AD-A044 906 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 9/2
MINOS. A LARGE-SCALE NONLINEAR PROGRAMMING SYSTEM (FOR PROBLEMS--ETC(U)
FEB 77 B A MURTAGH, M A SAUNDERS N00014-75-C-0865
UNCLASSIFIED SOL-77-9 NL

1 OF 2
AD
A044 906

MINOS

A LARGE-SCALE NONLINEAR PROGRAMMING SYSTEM

(For Problems With Linear Constraints)

USER'S GUIDE

BY

BRUCE A. MURTAGH AND MICHAEL A. SAUNDERS

TECHNICAL REPORT SOL 77-9

FEBRUARY 1977

Systems Optimization Laboratory

Department of Operations Research

Stanford University

Stanford California 94305

6

MINOS,

A LARGE-SCALE NONLINEAR PROGRAMMING SYSTEM

(For Problems With Linear Constraints).

USER'S GUIDE.

by

10 Bruce A. Murtagh[†] [redacted] Michael A. Saunders[‡]

9 TECHNICAL REP[redacted]T, 14 SOL-77-9

Feb[redacted]977 11

12 132 p.

SYSTEMS OPTIMIZATION LABORATORY

DEPARTMENT OF OPERATIONS RESEARCH

Stanford University
Stanford, California

15 N00014-75-C-0865, EY-76-S-03-0326

[†]School of Mechanical and Industrial Engineering, The University of New South Wales, Kensington, N.S.W., Australia.

[‡]On leave from Applied Mathematics Division, D.S.I.R., Wellington, New Zealand.

Research and reproduction of this report were partially supported by the Energy Research and Development Administration Contract EY-76-S-03-0326 PA #18; the Office of Naval Research Contract N00014-75-C-0865; and the National Science Foundation Grant MCS 76-20019 (formerly MCS 75-04544-A01).

408 765

[illegible]

TABLE OF CONTENTS

SECTION PAGE

I. Introduction

I.1. The Buzz Words

MINOS stands for "a Modular In-core Nonlinear Optimization System." It sounds roughly like the first character in "$-\infty$" (minus infinity), and its purpose is to optimize your own special objective function $f(x)$ by finding a point x which makes $f(x)$ as close to $\pm\infty$ as possible.

Since it is not normally meaningful to go quite that far, MINOS allows you to restrict the variables x to some feasible region specified by a set of linear constraints $(Ax \overset{\geq}{\underset{\leq}{=}} b)$ and a set of upper and lower bounds $(\ell \leq x \leq u)$. In particular, the matrix A may be large and sparse. Of course A may be small and sparse, or even small and dense; in general, the smaller and sparser the better.

Ideally your objective function $f(x)$ should be smooth, the nonlinear part of it should involve as few of the components of x as possible, and you should know how to compute the gradient vector $g(x) = (\partial f/\partial x_j)$ at any feasible point x.

With the important words thus introduced, we will now leave an example problem to speak for itself. The remainder of this section contains the following:

--a statement of the problem;

--a Fortran subroutine to compute $f(x)$ and $g(x)$;

--input cards to be read by MINOS;

--the output produced during solution of the problem.

The reader should not expect to understand all items immediately. However, the example should help to determine if a particular problem is of the requisite form, and may serve as a useful reference thereafter. Note that "x" in this example is the vector (x, y, z).

Example 1

Objective:	minimize $f(x, y, z) = x^2 + (y - 1)^4 - 4x + z$
Linear Constraints:	$x + y \leq 2$
	$x + y - z = 0$
Bounds:	$x \geq 0, \quad 1 \leq z \leq 5$
Nonlinear Variables:	x and y

Fortran Subroutine to Compute the Nonlinear Part of $f(x, y, z)$ *and its Gradient*

```
      SUBROUTINE CALCFG( MODE,N,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      DIMENSION  X(N),G(N)
C
C     EVALUATE FUNCTION  F   AND GRADIENT   G   AT CURRENT POINT
C     THE INTEGER  N   WILL BE  2   IN THIS EXAMPLE.
C
      T     = X(2) - 1.0
      F     = X(1)**2 + T**4
      G(1)  = 2.0*X(1)
      G(2)  = 4.0*T**3
      RETURN
      END
```

(The parameters of subroutine CALCFG are defined in Section III.3.)

Input Cards to be Read by MINOS (SPECS data, followed by MPS data)

```
BEGIN SPECS FOR EXAMPLE PROBLEM
   MINIMIZE
   OBJECTIVE                  OBJ.ROW
   RHS                        RHS1
   BOUNDS                     BND1

   ROWS                         10    * These 3 numbers must be
   COLUMNS                      20    * over-estimates of the dimensions
   ELEMENTS                    100    * of the constraint matrix

   ITERATIONS                  100
   SOLUTION                    YES

   PROBLEM NO.                   1
   NONLINEAR VARIABLES           2
   SUPERBASICS                   3
   LINESEARCH TOLERANCE        0.1
END EXAMPLE SPECS
```

```
NAME          EXAMPLE1
ROWS
 N  OBJ.ROW
 L  ROW001
 E  ROW002
COLUMNS
    X         OBJ.ROW      -4.0        ROW001        1.0
    X         ROW002        1.0
    Y         ROW001        1.0        ROW002        1.0
    Z         OBJ.ROW       1.0        ROW002       -1.0
RHS
    RHS1      ROW001        2.0
BOUNDS
 FR BND1      Y
 LO BND1      Z             1.0
 UP BND1      Z             5.0
ENDATA
```

Column numbers for fields on MPS data cards:

```
||  | NAME |  | NAME |  |  VALUE    |   | NAME |  |  VALUE     |
12  5     12  15    22  25         36   40    47  50          61
```

NOTE: The nonlinear variables X and Y must appear <u>first</u>

in the COLUMNS section of the MPS data.

Output from MINOS

```
        M I N O S     ---     VERSION  3.1    FEB  1977
        = = = = =


PROBLEM SPECIFICATIONS
----------------------
           BEGIN SPECS FOR EXAMPLE PROBLEM
              MINIMIZE
              OBJECTIVE                OBJ.ROW
              RHS                      RHS1
              BOUNDS                   BND1

              ROWS                       10      * THESE 3 NUMBERS MUST BE
              COLUMNS                    20      * OVER-ESTIMATES OF THE DIMENSIONS
              ELEMENTS                  100      * OF THE CONSTRAINT MATRIX

              ITERATIONS                100
              SOLUTION                  YES

              PROBLEM NO.                 1
              NONLINEAR VARIABLES         2
              SUPERBASICS                 3
              LINESEARCH TOLERANCE      0.1
           END EXAMPLE SPECS


PARAMETERS
----------
MPS INPUT DATA.
ROW LIMIT......        10     COLUMN LIMIT...        20     ELEMENT LIMIT..       100     AIJ TOLERANCE..  1.00E-10
LISTING LIMIT..         0     ERROR MSG LIMIT        10     LOW BND DEFAULT  0.0          UPP BND DEFAULT  1.00E+30

FILES.
INPUT MPS DATA.         5     SOLUTION FILE..         0     OLD BASIS MAP..         0     NEW BASIS MAP..         0
INSERT.........         0     PUNCH..........         0     LOAD NAMES.....         0     DUMP NAMES.....         0
(SCRATCH)......         8

FREQUENCIES.
CHECK ROW ERROR        25     FACTOR (INVERT)        50     SAVE BASIS MAP.       100     LOG ITERATIONS.         1

LP PARAMETERS.
ITERATIONS.....       100     PARTIAL PRICE..         1     MULTIPLE PRICE.         0     WEIGHT ON OBJ..  0.0
TARGET OBJ VAL.  0.0          ALIGNMENT TOL..   1.00000

NONLINEAR PROBLEMS.
PROBLEM NO.....         1     NONLINEAR VARS.         2     NONLINEAR FCNS.         1     DERIVATIVE LVL.         1
SUPERBASICS....         3     HESSIAN SIZE...         3     CONJ-GRD MODE..         1     VERIFY.........        NO
LINESEARCH TOL.   0.10000     REDUCED-GRD TOL   0.20000

MISCELLANEOUS.
PRINT LEVEL....         0     DEBUG LEVEL....         0     IMBED..........       YES
LU ROW TOL.....   0.00100     LU COL TOL.....   0.10000     LU MOD TOL.....   0.99000
```

```
INPUT LISTING
-------------
      1      NAME           EXAMPLE1
      2      ROWS
      6      COLUMNS
     11      RHS
     13      BOUNDS
     17      ENDATA


NAMES SELECTED
--------------
OBJECTIVE:    OBJ.ROW  (MIN)        1
RHS:          RHS1                  1
RANGES:                             0
BOUNDS:       BND1                  3


MATRIX STATISTICS
-----------------
                 TOTAL      NORMAL        FREE       FIXED     BOUNDED
ROWS:                3           1           1           1           0
COLUMNS:             4           1           1           1           1

NO. OF MATRIX ELEMENTS:          8    DENSITY:      66.667
NO. OF REJECTED COEFFS:          0    AIJTOL:  1.00000E-10
BIGGEST AND SMALLEST COEFFS:  1.00000E+00   1.00000E+00   (EXCLUDING OBJ AND RHS)

LENGTH OF HASH TABLE, COLLISIONS:  101       0

NO. OF  INITIAL  BOUNDS PROCESSED:           0
NO. OF SUPERBASICS SPECIFIED:                0

PARTIAL PRICE FACTOR, PARTITION:      1      3

NUMBER OF WORDS OF CORE AVAILABLE:       11776
```

```
ITERATIONS
----------

CRASH OPTION    1
FREE LOGICALS AND STRUCTURALS IN BASIS:       1       1

FACTORIZE      1    DEMAND        0    ITERATION      0    INFEAS       0    OBJECTV   0.0
SLACKS         2    LINEAR        0    NONLINEAR      1    ELEMS        4    DENSITY  44.4
P4 BUMPS       0    SPIKES        0    CORE REQD    153    L LIMIT   2818    U LIMIT  5636
LU BUMPS       0    SPIKES        0    AIJ ELEMS      2    L ELEMS      2    U ELEMS     1    F ELEMS      0   0.0
ITN     0 -- INFEASIBLE.  NUM =     1    SUM =    1.0000000000D+00

 ITN PH PP NOPT  DJ/RG   +SBS -SBS  -BS    STEP     PIVOT   NSPK     L       U NINF  SINF/OBJECTIVE     NFG  NSB  R1M H-CONDN CONV
   1  1  1   2 -1.00D+00    3    3    7  1.00D+00 -1.00D+00   0      2       1    1  1.0000000000D+00     0    0  0 0 0.0      FFFT

ITN     1 -- FEASIBLE SOLUTION.   OBJECTIVE =    3.0000000000D+00

   2  3  1   2 -8.00D+00    1    0    0  2.00D-01  0.0          1      2       2    0 -1.709775950+00     4    1  4 4 1.00D+00 FFTF
   3  4  1   2  2.00D-06    0    0    0  8.00D-01  0.0          1      2       2    0 -1.710726580+00     6    1  4 4 1.00D+00 TTTT
   4  3  1   1 -1.80D-01    6    0    0  1.00D+00  0.0          1      2       2    0 -1.722423930+00     8    2  4 4 4.60+00 TTTF

RG TOLS REDUCED.   TOLRG =     7.071D-07

   5  4  1   0  1.50-03     0    0    0  1.00D+00  0.0          1      2       2    0 -1.722470200+00     9    2  4 4 5.00D+00 FFFF
   6  4  1   0  1.60-05     0    0    0  1.00D+00  0.0          1      2       2    0 -1.722470390+00    10    2  4 4 4.90+00 FFFF
   7  4  1   0  2.20-07     0    0    0  1.00D+00  0.0          1      2       2    0 -1.722470390+00    11    2  4 4 4.80+00 FFTF
   8  4  1   0  1.50-09     0    0    0  1.00D+00  0.0          1      2       2    0 -1.722470390+00    12    2  4 4 4.80+00 TTTT

BIGGEST DJ =      0.0           NORM RG =     1.461D-09   NORM PI =      7.071D-01

EXIT -- OPTIMAL SOLUTION FOUND.

NO. OF ITERATIONS:                      8        PHASE:    3         OBJECTIVE:    -1.722470393711D+00

NO. OF FUNCTION AND GRADIENT CALLS:     12
```

```
PROBLEM NAME:    EXAMPLE1             OBJECTIVE VALUE:   -1.72247039370+00

STATUS:          OPTIMAL SOLN         PHASE:          3   ITERATION:      8

OBJECTIVE:       OBJ.ROW  (MIN)
RHS:             RHS1
RANGES:
BOUNDS:          BND1

SECTION 1 - ROWS

 NUMBER  ...ROW..  AT  ...ACTIVITY...  SLACK ACTIVITY ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY   ..I

      5  OBJ.ROW   BS       -4.12996        4.12996          NONE           NONE        -1.00000     1
      6  ROW001    SBS       1.87004        0.12996          NONE          2.00000       0.00000     2
      7  ROW002    EQ        0.0            0.0          0.0            0.0            -1.00000     3

SECTION 2 - COLUMNS

 NUMBER  .COLUMN.  AT  ...ACTIVITY...  .OBJ GRADIENT. ..LOWER LIMIT.  ..UPPER LIMIT.  .REDUCED COST.   M+J

      1  X         SBS       1.50000       -1.00000    0.0                NONE          0.0          4
      2  Y         BS        0.37004       -1.00000          NONE           NONE          0.0          5
      3  Z         BS        1.87004        1.00000       1.00000        5.00000          0.0          6
      4  RHS1      EQ       -1.00000        0.0          -1.00000       -1.00000          0.0          7
```

I.2. Formal Introduction

MINOS is a computer program designed to minimize a linear or nonlinear function subject to linear constraints.[†] In algebraic terms it is designed to solve problems of the form

$$P: \quad \begin{array}{ll} \text{minimize} & f(x) + c^T x \\ \text{subject to} & Ax \gtreqqless b, \qquad \ell \le x \le u \end{array}$$

given that $f(x)$ is a continuous differentiable function with gradient $\nabla f(x) = [\partial f/\partial x_j] = g(x)$.

Such problems range from small, unconstrained optimization problems through constrained linear least squares (data-fitting) models to large-scale linear programs with (or without) nonlinear terms in the objective. In general the constraint matrix A is assumed to be large and sparse.

Fundamental to the system is an efficient and reliable implementation of the revised simplex method for linear programming (LP).[‡] This combines established sparse-matrix technology with stable numerical methods for computing and modifying a triangular factorization of the usual square basis matrix B.[‡‡] The code is intended to be suitable for production runs on purely linear problems. The data formats used for various files therefore follow those adopted by current commercial mathematical programming systems. In particular, the

[†] The theory of the solution technique is described in a companion paper, Murtagh and Saunders (1976).

[‡] Dantzig (1963).

[‡‡] Bartels and Golub (1969), Bartels (1971), Saunders (1976).

quantities A, b, c, ℓ and u are defined in the format accepted by the IBM Mathematical Programming Systems MPS/360, MPSX and MPSX/370. Ideally the reader of this manual should have experience in using such systems and should (a fortiori) be familiar with the concepts and terminology associated with the simplex method.

Recall that the <u>basic</u> variables associated with B may take solution values anywhere between their upper and lower bounds, and that the remaining variables are defined to be <u>nonbasic</u> at one or other of their bounds. The simplex method always works with solutions of this form. In order to extend the simplex method to problem P we introduce a new class of variables, which will be called <u>superbasic</u>. Both basic and superbasic variables may vary between their bounds, but in the reduced-gradient† approach used here their roles are different. Suppose there are ns superbasic variables at some stage. These are essentially free to move in any desirable direction, namely one which will improve the value of the objective functions; in effect they provide the driving force. The basic variables can then be adjusted so that the full set of variables x remains feasible with respect to the linear constraints. If it appears that no improvement can be made with the current set of superbasics, one (or more) of the nonbasic variables is selected to become superbasic, ns is increased and the process is repeated. At all stages, if a basic or superbasic variable encounters one of its bounds, an adjustment occurs in which that variable is made nonbasic and the total number of superbasics is reduced by 1.

---

†Wolfe (1962, 1967).

Users familiar with LP may interpret the simplex method as being exactly the above process, with ns oscillating between 0 and 1.

At any given stage, the number of superbasics (ns) measures in some sense how far the current point x is from a vertex of the simplex (polyhedron) defining the feasible region. In other words, x lies somewhere on a face of the polyhedron, which defines a subspace of dimension ns. An important part of MINOS is a stable implementation of a quasi-Newton (or variable-metric) method for optimizing the superbasic variables within each appropriate subspace. This method uses a triangular matrix of dimension ns to approximate the reduced Hessian (i.e., a suitably transformed sub-section of the matrix of second derivatives, $[\partial^2 f/\partial x_i \partial x_j]$). Good rates of convergence are usually achieved.

If the number of superbasic variables ever grows as large as 100 or 200 the storage required for the quasi-Newton procedure starts to become excessive. In such cases MINOS will commence using a conjugate-gradient method,[‡] which requires very little storage. The rate of convergence normally drops severely, but at present this is the only practical alternative for problems with very many nonlinearities.

The concept of superbasic variables can be valuable even for purely linear problems. As an example, the solution to a particular problem may provide a feasible but non-vertex starting point for a modified version of the problem (e.g., if the bounds were altered on some nonbasic variables). The re-starting features in MINOS allow

[†] Davidon (1959), Fletcher and Powell (1963).

[‡] Fletcher and Reeves (1964).

the user to take advantage of such situations easily. On a less routine level, MINOS also provides a framework within which experiments can be performed on large problems using non-simplex steps, perhaps with a view to reducing the total number of iterations required.

I.3. The Canonical Form

Without loss of generality, problem P may be re-cast in the form

$$\bar{P}: \quad \begin{array}{ll} \text{minimize} & f(x_N) - s_{obj} \\ \text{subject to} & [A_N \; A_L \; b \; I] \begin{bmatrix} x_N \\ x_L \\ \rho \\ s \end{bmatrix} = 0, \qquad \ell \le \begin{bmatrix} x_N \\ x_L \\ \rho \\ s \end{bmatrix} \le u \\ \text{given} & \nabla f(x_N) = g(x_N) \end{array}$$

where the following definitions hold:

x_N = the <u>nonlinear</u> <u>variables</u>, i.e., those that are directly involved in the function $f(x_N)$.

x_L = the <u>linear</u> <u>variables</u>, i.e., the remaining part of x.

ρ = the <u>right-hand side</u> variable, which has bounds $\ell_\rho = u_\rho = -1.0$

s	=	the <u>slack</u> or <u>logical variables</u> (one for each row of A).
s_{obj}	=	the slack variable corresponding to the linear part (if any) of the objective function. The term $c^T x$ in problem P is thus imbedded in the constraint matrix as the "obj"-th constraint, $c^T x + s_{obj} = 0$, with s_{obj} being a free variable. (This row may be omitted if $c = 0$ or if c is included in f.)
A	=	$[A_N \; A_L]$
m	=	the number of rows in A.
n	=	the number of columns in A.
nn	=	the number of nonlinear variables in x_n.
ns	=	the number of superbasic variables.

At any particular stage, the columns of $[A\ b\ I]$ are implicitly ordered as follows:

$$[A\ b\ I]\hat{P} = \begin{array}{|c|c|c|} \multicolumn{1}{c}{m} & \multicolumn{1}{c}{ns} & \multicolumn{1}{c}{n+1-ns} \\ \hline \quad B \quad & \quad S \quad & \quad N \quad \\ \hline \multicolumn{1}{c}{\text{Basics, as in simplex method}} & \multicolumn{1}{c}{\text{Superbasics}} & \multicolumn{1}{c}{\text{Nonbasics (variables at upper or lower bound)}} \end{array}$$

where $\hat{P}$ is a permutation. The nonlinear variables may end up anywhere in B, S or N, and similarly the superbasic variables in S may be linear or nonlinear. The only requirement is that the basis matrix B be nonsingular.

The above partitioning should be remembered when interpreting output from MINOS and when modifying certain basis files for input.

II. Nonlinear Problems

At any given stage, the reduced-gradient method regards the objective as an essentially unconstrained function of the superbasic variables alone. (Basic and nonbasic variables are eliminated by means of the active constraints.) It is therefore possible to use unconstrained optimization methods on some "reduced function" of the ns superbasic variables. Suppose that the original objective has gradient g and Hessian G (i.e., the matrix of second partial derivatives). The gradient and Hessian for the reduced function are then Z^Tg and Z^TGZ, where Z is defined in terms of the current basis as the $n \times ns$ matrix

$$Z = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cl} \left[\begin{array}{c} -B^{-1}S \\ I \\ 0 \end{array}\right] & \begin{array}{l} m \\ ns \\ n+1-m-ns \end{array} \\ ns & \end{array} .$$

Note that it would not be practical to compute Z or Z^TGZ (assuming m to be large and ns to be more than 1 or 2). Any unconstrained optimization method to be used must therefore make do with vectors of the form Z^Tg or Zp, which can be computed with relative ease.

II.1 The Quasi-Newton Method

Whenever storage permits, as specified by a HESSIAN DIMENSION card in the SPECS file (see Section IV), MINOS maintains a quasi-Newton approximation to the reduced Hessian thus:

$$R^T R \approx Z^T G Z$$

where R is a dense, nonsingular, upper triangular matrix of dimension ns. This normally provides superlinear convergence to a minimum within the subspace defined by each set of superbasic variables. The factorized form $R^T R$ is justified by the fact that $Z^T G Z$ must be positive definite (or at least semi-definite) at such a minimum. It also guarantees that the equations

$$R^T R s = -Z^T g, \qquad p = Zs$$

will generate a direction p along which the original objective will decrease (since the projection of p along the gradient is $p^T g = s^T Z^T g = -s^T R^T R s = -\|Rs\|_2^2 < 0$).

Various modifications are made to R each iteration to account for new curvature information and to allow for changes in the number of superbasics. For example, when a nonzero step is taken, a new matrix $\bar{R}$ is computed in two stages to satisfy

$$\bar{R}^T \bar{R} = R^T R + vv^T - ww^T$$

for some vectors v and w (which are determined by the so-called "Complementary DFP" or "BFGS" formula, e.g., see Broyden (1972)). The result of such operations on R is briefly indicated in the iteration log by two integers labelled "R1M" (short for Rank-one Modifications). In general, these numbers are:

positive	if the modifications were successful;
zero	if the change in reduced gradient was too small for v and w to be meaningful (in which case R is unaltered);
negative	if numerical error prevented a particular modification.

Another useful quantity appearing in the iteration log is an estimate of the "condition number" of R^TR. See the discussion of H-CONDN in Section VII ; also see Section II.3.

II.2. The Conjugate-Gradient Method

If insufficient storage has been allocated for R, MINOS will discontinue the quasi-Newton method for generating search directions and switch to a conjugate-gradient method (which does not need R). The dimension of R may be specified separately from the upper limit on superbasics, thus:

```
SUPERBASICS          200
HESSIAN DIMENSION    100
```

In this example a conjugate-gradient method will be used if the number of superbasics exceeds 100. Quasi-Newton iterations can be suppressed completely by the card

```
HESSIAN DIMENSION      0
```

(The total no. of iterations will probably increase significantly.)

Five versions of "CG" are currently available via the SPECS card

```
CONJUGATE GRADIENT VERSION    k
```

for the following values of k:

k	Method
1	Fletcher and Reeves (1964)
2	Polak and Ribiere (1969)
3	Perry (1976)
4	Memoryless DFP
5	Memoryless Complementary DFP

The general form of the first three methods is

$$s = -Z^T g + \beta s_{old}$$

where s and s_{old} are the new and previous search directions for the superbasics, and β depends on the method. The other two methods are derived from the corresponding quasi-Newton formulas, taking the initial reduced Hessian approximation to be I. They add to s a term of the form

$$\gamma(Z^T g - Z^T g_{old}) .$$

Restarts $(\beta = \gamma = 0)$ occur at every change to the basic or superbasic sets, or after ns consecutive unconstrained steps with a particular set of ns superbasics.

The state of research on conjugate-gradient methods is such that no clear recommendation can be made for the choice of k. If an accurate linesearch is used, all methods are rather similar and $k = 1$ or 2 should be tried. If the objective function is so expensive to evaluate that an inaccurate search is necessary, $k = 3$ or 5 may be preferable. Method 4 appears from limited experience to be the least successful. Considerable further research is needed in this area.

II.3. Scaling Nonlinear Variables

No rigid rules can be given here that would apply usefully to all nonlinear functions, but one quantity in particular may be worth examining if it seems that optimization is proceeding very slowly. This quantity is cond(G), the condition number of the Hessian, which may be very sensitive to scaling.

We shall illustrate with a particular function of two variables, viz.

$$f(x, y) = x^b y^c$$

where b and c are known parameters. It is readily verified that the Hessian for f is

$$G(x, y) = \begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y \\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2 \end{bmatrix} = \frac{f}{x^2 y^2}\begin{bmatrix} b(b - 1)y^2 & bcxy \\ bcxy & c(c - 1)x^2 \end{bmatrix} .$$

Furthermore, G may be factorized as

$$G(x, y) = \frac{f}{x^2 y^2} LL^T$$

where

$$L = \begin{bmatrix} \beta y & \\ \gamma x & \delta x \end{bmatrix}$$

$$\beta^2 = b(b - 1)$$

$$\gamma = bc/\beta$$

$$\delta^2 = c(1 - b - c)/(b - 1).$$

A triangular factorization of this kind provides a lower bound on the condition number of G, in terms of the ratio of the largest and smallest diagonals of L. In this case, suppose that $|\beta y| \geq |\delta x|$ at the optimal solution. Then

$$\text{cond}(G) = \text{cond}(L)^2 \geq \left(\frac{\beta y}{\delta x}\right)^2 .$$

Assuming that β and δ are reasonable numbers (of order 1), it appears that the ratio y^2/x^2 at the optimal solution may be a useful indicator of the sensitivity of the optimum to small changes in the data. To make the estimate of $\text{cond}(G)$ close to 1, x and y should be scaled to have approximately the same values at the solution. (The same conclusion would be drawn if $|\beta y| \leq |\delta x|$.)

Warning: In the reduced-gradient method, scaling for the above reasons should not be at the expense of upsetting the scale of the constraint matrix, A. This is because the quantity of interest is really $\text{cond}(Z^TGZ)$ rather than $\text{cond}(G)$, where Z is a matrix whose scale (and condition) depend on A.

Note that if certain variables (and hence columns of A) are scaled to improve $\text{cond}(G)$, it may be necessary to scale certain rows of A to maintain the condition of A and Z^TGZ. It may then be necessary to scale certain other columns of A, and so on. Clearly, it is preferable to be careful with scaling during the initial formulation of a problem.

Finally, the analysis suggested above will usually be applicable to certain obvious rows and columns of G, rather than to G as a whole; for example, if G is block-diagonal, each block may be analyzed in turn.

III. Data Required from the User

The following three items must be supplied by the user:

1.	the SPECS file	to specify certain run-time parameters;
2.	the MPS file	to specify the linear constraints in standard MPS format;
3.	subroutine CALCFG	to compute the nonlinear part of the objective function and its gradient.

The user may also (optionally) supply

4.	certain BASIS files	to define an initial solution.

Since several or all of the various input files could be part of the card input stream, it is important to note that MINOS reads data in the following order:

(a) SPECS data

(b) MPS data

(c) BASIS data

(d) any data required by subroutine CALCFG.

III.1. The SPECS File

Certain parameters may be set by the user via a list of "problem specifications," hereafter called SPECS. These are assumed to be a deck of 80-character card images of the form

```
BEGIN SPECS FOR FRED
          .
          .
          (list of keywords and values)
          .
          .
END FRED
```

where the words BEGIN and END are themselves keywords. (Other characters on BEGIN and END cards are ignored.) The SPECS file must exist; it is normally the first data set in the card input stream (file 5).

Broadly speaking, each card in the SPECS file contains a sequence of items punched in free format (i.e., separated by at least one blank or =). The items selected from each card are

1. The first word (the keyword; only the first 3 characters are significant);
2. The second word (8 characters, treated as two sets of 4);
3. The first number (up to 8 characters; either integer or real).

In the following examples the significant characters are underlined:

```
OBJECTIVE              PROFIT
ROWS                      500
AIJTOL                 1.0E-6
LOWER BOUND              -1.0
SOLUTION FILE              20
OLD BIT MAP ON FILE         9
```

Only occasionally is the second word meaningful (e.g., PROFIT and FILE above). Words such as BOUND and BIT MAP are extracted but later ignored.

Blank cards are allowed, and comments (i.e., any characters) may occur after an asterisk, e.g.,

```
NONLINEAR VARIABLES          20     * X005, X010, ..., X100.
```

Default Values

Parameters that are not specified in the SPECS file assume certain default values, most of which should be acceptable for normal use. In the following list the default values are given for all legal keywords. Where possible (except for some of the items printed in lower case) the list itself constitutes a valid SPECS file. Note, however, that its effect would be essentially the same as the file

```
BEGIN
END
```

(which is also valid).

A SPECS File Showing all Legal Keywords and Their Default Values

```
BEGIN  SPECS FILE SHOWING DEFAULT VALUES FOR KEYWORD PARAMETERS
*
*
*  KEYWORDS FOR MPS DATA
*
*
   MINIMIZE
   OBJECTIVE   =    the first name encountered
   RHS         =                 ditto
   RANGES      =                 ditto
   BOUNDS      =                 ditto

   ROWS                           100      *
   COLUMNS                        300      *  (or  3xROWS)
   ELEMENTS (COEFFICIENTS)       1500      *  (or  5xCOLS)
   AIJ ZERO TOLERANCE         1.0E-10      *
   LOWER BOUND                    0.0      *
   UPPER BOUND                1.0E+30      *
                                           *
   INPUT FILE                       5      *  (MPS data)
   LIST LIMIT                       0      *  (for printing of MPS data)
   ERROR MESSAGE LIMIT             10      *  (during MPS input)
*
*
*
*  KEYWORDS FOR THE SIMPLEX ALGORITHM
*
*
   CRASH OPTION                     1      *  (Triangular CRASH on all vars)
   ITERATIONS                     300      *  (3xROWS + 10xNONLINEARS)
   PARTIAL PRICE                    1      *
   MULTIPLE PRICE                   0      *  (BEWARE - not like normal LP)
                                           *
   WEIGHT ON OBJECTIVE            0.0      *  (during Phase 1)
                                           *
   FACTORIZATION FREQUENCY         50      *  (i.e. INVERT FREQUENCY)
   CHECK FREQ                      25      *  (test on row residuals)
   SAVE  FREQ                     100      *  (for BIT MAP)
   LOG   FREQ                       1      *  (for ITERATION LOG)

   SOLUTION    IF OPTIMAL, INFEASIBLE, OR UNBOUNDED
*
*
*
*  FILES FOR SAVING AND LOADING BASIS DESCRIPTIONS
*
*
   OLD BASIS FILE                   0      *  (BIT MAP)
   NEW BASIS FILE                   0      *  (BIT MAP)
   INSERT    FILE                   0      *  (Industry standard)
   PUNCH     FILE                   0      *  (for saving of INSERT data)
   LOAD      FILE                   0      *  (NAMES and VALUES)
   DUMP      FILE                   0      *  (for saving of  LOAD  data)
   SOLUTION  FILE                   0      *  (Don't confuse with the
                                           *  printed solution)
```

```
*
*
*
*  PARAMETERS FOR NONLINEAR PROBLEMS
*
*
   PROBLEM NUMBER                   0      *  (Parameter NPROB for CALCFG)
   NONLINEAR VARIABLES              0      *  (Default is pure LP)
   SUPERBASICS LIMIT                1      *  (or  NONLINEARS + 1)
   HESSIAN DIMENSION                1      *  (min( SUPERBASICS, 30 ))
   CONJUGATE GRADIENT VERSION       1      *  (FLETCHER-REEVES)
                                           *
   VERIFY                          NO      *  (Suppress gradient check)
                                           *
   LINESEARCH TOLERANCE          0.01      *  (Must be between 0.0 and 1.0)
   REDUCED-GRADIENT TOL           0.2      *  (ditto)
*
*
*
*  MISCELLANEOUS
*
*
   DEBUG LEVEL                      0      *
   PRINT LEVEL                      0      *
   IMBED                          YES      *  (Store part of LU file in A)
                                           *
   LU ROW TOLERANCE             0.001      *  (for initial LU factorization)
   LU COL TOL                     0.1      *  (ditto)
   LU MOD TOL                    0.99      *  (for updating LU factors)
*
*
*
*  THE FOLLOWING KEYWORDS EACH ENABLE A CERTAIN OPTION.
*  THEY SHOULD NOT APPEAR IF THE OPTION IS NOT WANTED.
*
*
   ALIGNMENT TOLERANCE            1.0      *  (Experimental, for pure LP)
   CALL CALCFG IF OPTIMAL                  *  (One last entry with NSTATE=2)
   TARGET OBJECTIVE VALUE         0.0      *  (Experimental, for pure LP)

END LIST OF DEFAULT VALUES


NOTE:  The keywords  BEGIN  and  END  are required.
```

III.2. The MPS File

After the SPECS file has been successfully read, the data specifying the constraint matrix is input in standard MPS format. This format is defined under the title "CONVERT DATA" in IBM document number SH20-0968-1, "Mathematical Programming System-Extended (MPSX), and Generalized Upper Bounding (GUB)," pp. 199-209. Alternatively, see IBM document number H20-0476-2, "Mathematical Programming System/360 Version 2, Linear and Separable Programming -- User's Manual," pp. 141-151.

The various sections of the data must be in the following order:

NAME
ROWS
COLUMNS
RHS
RANGES
BOUNDS
ENDATA

Notes on MPS Data

1. The relevant column numbers and contents for the fields on each data card are as follows:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
2-3	5-12	15-22	25-36	40-47	50-61
code	name	name	value	name	value

2. The NAME card may define a name of any length, beginning in column 15. Only the first 8 characters are saved for printing with the solution.

3. The ROWS section may define constraints to be of type $=$, $\geq$, $\leq$ or free, using the characters E, G, L or N respectively (in column 2 or column 3). It is not essential to have an objective row; thus there may be 0, 1 or more rows of type N.

4. The COLUMNS section may define 1 or 2 matrix elements per card. For nonlinear columns, it is also acceptable to have 0 elements (i.e., just the name of the variable).

   NOTE: Nonlinear variables must occur first in the COLUMNS section, ordered in a manner that is consistent with the array X in the user-written subroutine CALCFG.

5. The RHS section may contain several RHS's. A specific one may be requested via the specifications, otherwise the first RHS is taken.

6. Similarly for the RANGES section.

7. Similarly for BOUNDS. The types of bound allowed are UP, LO, FX, FR, MI, PL.

   Warning: If a variable is essentially unconstrained the bound type FR (free) should always be used. Never use a large negative LO bound. Similarly if a variable XI has bounds $-\infty \leq XI \leq 5$ (say), then 2 cards are required as follows:

```
FR BND1       XI
UP BND1       XI             5.0
```

8. The INITIAL bounds set. The name INITIAL is reserved to specify (optionally) a special bounds set which can be used to assign initial values to nonlinear variables. The INITIAL bounds set must appear after any normal bounds sets (if any). A warning is given if it is the first set encountered after the BOUNDS card. The following example illustrates the bound types allowed:

Field 1	Field 2	Field 3	Field 4
FX	INITIAL	XN1	10.0
LO	INITIAL	XN2	
UP	INITIAL	XN3	

In this example:

(a) The nonlinear variable XN1 will be initialized at value 10.0. (Its initial state will be superbasic.) If 10.0 lies outside the bounds on XN1, the initial value will be the nearest bound.

(b) Nonlinear variables XN2 and XN3 will be initialized at their lower and upper bound values respectively. (Their initial state will be nonbasic.)

(c) Any nonlinear variables that are not specified directly will initially be nonbasic at their smallest bound (in absolute value).

Note: The INITIAL bounds set provides an important facility if used in conjunction with CRASH OPTION 2 (which prevents nonlinear variables from being selected for the initial basis). By construction, all nonlinear variables will then be either superbasic or nonbasic, and they will be held fixed at their initial values while the remaining (linear) variables are optimized by a normal Phase 1/Phase 2 simplex procedure. Once it has been determined that the associated linear program is either optimal or infeasible, the nonlinear variables will be allowed to vary and optimization will continue in normal reduced-gradient mode.

It follows that an INITIAL bounds set and CRASH OPTION 2 should be used if good starting values are known for all of the nonlinear variables. (At least one FX indicator must be used to achieve the above effect, and the SUPERBASICS card should specify a suitably large number. If FX is used for all nonlinear variables, then CRASH OPTION 2 may be omitted.)

If a starting point is specified by any other means (i.e., via OLD BASIS, INSERT BASIS or LOAD BASIS cards), the INITIAL bounds set will be ignored. This allows the user to re-start a run without altering the MPS file.

Warnings, provisos, etc.

1. The constraint matrix is not scaled by the program, so the user should try to ensure that

   a) all elements a_{ij} are reasonably close to 1.0;

   b) units are chosen so that all x_j satisfy $10^{-2} \leq |x_j| \leq 10^2$ or $x_j = 0$.

   These are by no means absolute requirements but are recommended for numerical reliability (and to ensure that there are at least some significant figures in the printed solution).

2. The program <u>does</u> make allowance for the scale of the objective function and, to a lesser extent, the RHS. The sizes of the objective and b_i elements are therefore somewhat less critical.
3. The nonlinear objective function is evaluated only at feasible points. The BOUNDS section should in general include bounds on all nonlinear variables so that "feasible points" are indeed away from singularities in the objective and its gradient, e.g., if $f(\underset{\sim}{x}) = 1/x_1 x_2^2$ and it is known that $x_i \approx 10$ at the solution, it would be advisable to include bounds forcing $x_i \geq 1$ (say).

III.3. <u>Subroutine CALCFG</u>

<u>Computation of the Objective Function and its Gradient</u>

Let $f(x)$ be the nonlinear part of the objective function, with corresponding gradient vector $g(x)$. Both $f(x)$ and $g(x)$ must be computed by a subroutine called CALCFG, whose specification and parameters are as follows. (Recall that any linear terms in the objective function may be included as part of $f(x)$ or as a free row in the constraint matrix A.)

<u>Specification</u>

```
      SUBROUTINE CALCFG (MODE, N, X, F, G, NSTATE, NPROB)
      IMPLICIT  INTEGER(I-N), REAL(A-H, O-Z)
      DIMENSION  X(N), G(N).
C
C     On machines with short word length, e.g., IBM, Univac, it is
C     necessary to use double precision floating-point.  Hence
C      IMPLICIT  INTEGER(I-N), REAL*8(A-H, O-Z)
```

Parameters

MODE	(Input, output) A control parameter defining the type of function and derivative information required. On entry: MODE = 2 means the user must assign the function value $f(x)$ to F and the gradient values $g(x)$ to G(j), $j = 1, 2, \ldots, N$, using the values of X stored in X(j), $j = 1, 2, \ldots, N$. (Other values of MODE not yet implemented.) On exit: If the user wishes to terminate solution of the current problem, MODE should be given a negative value (viz. -1).
N	(Input) The number of variables involved in $f(x)$. Note: These must be the first N variables in the constraint matrix A. In general, A will be of dimension m by n with $N < n$.
X	(Input) An array of dimension N containing the current values of the nonlinear variables x. Note: The contents of X should not be altered, except perhaps if NSTATE = 2 (see below).
F	(Output) The computed value of $f(x)$.
G	(Output) The computed gradient vector $g(x)$. The user must store the partial derivative $\partial f/\partial x_j$ in G(j), $j = 1, 2, \ldots, N$.

NSTATE (Input) A control parameter informing the user of the current state of the optimization process.

NSTATE = 0 if there is nothing special about the current entry.

NSTATE = 1 if this is the first entry to CALCFG. The user may wish to read data or set up certain local arrays for use in computing F and G during the present and subsequent entries to CALCFG.

NSTATE = 2 if the current solution in X is optimal. The user may wish to perform additional computations on X.

Notes about entry with NSTATE = 2

1. This will occur only if the user includes a card of the form

    CALL OBJECTIVE WHEN OPTIMAL

   in the problem specifications.

2. This will be the last entry to CALCFG.

3. The entry will not occur if iterations are terminated before optimality is recognized.

4. The entry will be made before the solution is output to the printer and/or the Solution File. Any changes to the nonlinear variables X will therefore appear in the solution output. (For example, one might wish to round the elements of X to the nearest integer.)

NPROB (Input) - An integer which can be set by the user via a card of the form

PROBLEM NUMBER 5

in the problem specifications. This parameter may be used to branch to one of several function calculations within CALCFG. The user may ignore it or use and change it as desired.

III.4. BASIS Files

Four distinct methods are available for loading and saving basis descriptions.† They are invoked by SPECS cards of the following form. (Just one option per card. The file numbers may be whatever the user chooses.)

Loading	Saving
OLD BASIS FILE 10	NEW BASIS FILE 11
INSERT FILE 20	PUNCH FILE 21
LOAD FILE 30	DUMP FILE 31
CRASH OPTION 0, 1 or 2	SOLUTION FILE 50

The keywords OLD, INSERT, LOAD are mutually exclusive. If more than one positive file number is specified, the order of precedence is that just given. If no starting files are specified, one of the CRASH options takes precedence.

The keywords NEW, PUNCH, DUMP, SOLUTION may all specify positive file numbers. Hence from zero to four files may be created at the end of a run, in the order just given. In addition, a NEW

†See also the INITIAL bounds set in Section III.2.

BASIS FILE may be created every k iterations during a run, where k is specified by a SAVE FREQUENCY card.

Normally the various basis files are used in the following circumstances. Fuller details are given in Section V.

1. OLD/NEW bit map.[†] This is the most compact and efficient method for starting from an earlier run on either the <u>same</u> problem or one of the same <u>dimensions</u>.

2. INSERT/PUNCH. This provides compatibility with certain commercial systems (e.g., MPS/360, MPSX/370, MPS III, APEX). The PUNCH file from a particular problem can be used as an INSERT file for restarting solution of the same problem. Occasionally it may be possible to modify the INSERT file and/or the problem and still obtain a useful advanced basis.

   The standard MPS format has been slightly generalized within MINOS to allow saving and reloading of nonbasic solutions.

3. LOAD/DUMP. This is similar in vein to INSERT/PUNCH but allows more direct specification of a list of basic and superbasic variables (both structurals and logicals) and their desired starting values. It is usually easier to ensure that a DUMP file (rather than a PUNCH file) is suitable for restarting a modified problem.

[†] So-called because each variable is represented by just one character.

4. SOLUTION file. This may be used to communicate solution values (e.g., x, π, objective gradient, bounds) to another program for further computation. It uses essentially the same format as that used for the printed solution. For greater precision it uses format 1PE16.6 for all values, rather than F16.5.

IV. Problem Specifications

IV.1. The SPECS file -- Syntax

Each card in the SPECS file must be of the form

⟨keyword⟩ ⟨identifier⟩ ⟨idlist⟩ ⟨number⟩ ⟨comment⟩

where items may be written in free format with one or more blanks or '=' s as separators. In general (depending on ⟨keyword⟩) not all items need be present.

⟨keyword⟩	A string of non-blank characters beginning anywhere in columns 1 through 70. A keyword is defined by its first 3 characters; any others are ignored.
⟨identifier⟩	A list of 8 consecutive characters, the first of which is any non-blank character except the following, which are hereby defined to be numeric characters:

0 1 2 3 4 5 6 7 8 9 + -

Valid use of identifiers:

```
OBJECTIVE = COST ROW
RHS       = ...Z.
RANGES    = RNG001
BOUNDS    = LOW1.E-5
```

Invalid use of identifiers:

```
OBJECTIVE = -Z
BOUNDS    = 2Q
BOUNDS    = 1.0E-7
```

Note that blanks may be imbedded in the last 7 characters, but the same blanks must then appear in the corresponding MPS data cards.

⟨idlist⟩ A list of items separated by blanks or '=' s. The first character of each item is not one of the numeric characters listed above. All items in the list are ignored.

Valid use of idlist:

WEIGHT ON OBJ (DURING PHASE ONE) = 100.0

Invalid use of an idlist:

WEIGHT ON OBJ (DURING PHASE 1) = 100.0

⟨number⟩ A list of up to 8 consecutive characters, the first of which is numeric. The number may be either an integer or a real constant. It will be treated as an integer if all remaining characters are numeric; it should then be typed in I format.

Valid use of integers:

ROWS = 2000

COLS = +2000

Invalid use of integers:

ROWS = 1.0E+3

A number will be treated as a real if any character other than the first is '.', 'E' or 'D'; it should then be typed in F, E or D format.

Valid use of reals:

```
AIJ TOLERANCE =  0.00001
WEIGHT ON OBJ = +1.0E+5
LOWER BOUND   = -1.0D+30
```

Invalid use of reals:

```
LOWER BOUND = .1E-5
```

(comment) Scanning of a specification card is terminated under the following conditions:

1. if the first non-blank character is an '*' (i.e., if the first character in the keyword is an '*'),
2. if any blank or '=' is followed by an '*'.
3. If an integer or a real has been recognized.

Hence a comment may be any characters following an '*' or an integer or a real.

Valid use of comments

```
*
OBJ              = COST 02 * 2ND ALTERNATIVE
ROWS 1000 (OR LESS)
WEIGHT ON OBJ = 10.0 IN PHASE 1
```

IV.2. The SPECS file -- Keywords

The following is an alphabetical list of recognized keywords. A typical use of each keyword is given, along with a definition of the quantities involved and various comments or warnings. In many cases, the value associated with a keyword is denoted by a letter such as k, and allowable values for k are subsequently defined.

● AIJ TOLERANCE 0.001

During input of the MPS file, matrix coefficients a_{ij} will be ignored if $|a_{ij}|$ is less than or equal to the specified tolerance.

● ALIGNMENT TOLERANCE 1.0

A parameter invoking a certain algorithmic option which affects the search direction used for superbasic variables. Specifically, a subset of the superbasics is selected such that each member x_j is close to its one of its bounds and has a significant gradient component moving x_j towards that bound. The search direction is scaled in a way that will cause all members of the subset to reach their bounds simultaneously. All members will therefore be "squeezed" out of the superbasic set, unless a basic variable happens to reach a bound first.

1. The ALIGNMENT keyword should not appear if alignment is not required.
2. Alignment is unlikely to be successful on nonlinear problems.
3. Use of the specified tolerance depends on the particular version of subroutine ALIGN.

● BOUNDS BOUND01

The 8-character name of the bound set to be selected from the MPS file.

1. BNDS is a valid alternative keyword.
2. If no BOUNDS keyword exists, or if the name specified is blank, the <u>first</u> bound set in the MPS file will be selected.
3. If the MPS file contains one or more bound sets, but the user wishes <u>no</u> bound set to be used, some dummy name should be specified, e.g., BOUNDS = NONE.

● CALL OBJECTIVE WHEN OPTIMAL

This requests a final call to the user's subroutine CALCFG when (and if) an optimal solution has been obtained.

1. Refer to the parameter "NSTATE" in Section III.3.
2. The CALL keyword should not appear if a final call is not required.

● CHECK FREQUENCY k

The residual vector $r = b - s - Ax$ will be computed every k-th iteration. Refactorization of the current basis B is requested if $\|D^{-1}r\|$ is too large, where $D = \text{diag}(d_i)$ and d_i is the largest element in the i-th row of B. Normally k should be about 25 or half the FACTORIZATION FREQUENCY.

● COEFFICIENTS 3000

See ELEMENTS.

● COLUMNS 1000

An over-estimate of the number of columns in the constraint matrix (excluding slacks). Default is 3m where m is specified by the ROWS card.

● CONJUGATE GRADIENT VERSION k

This specifies which version of the method of conjugate gradients should be used if there is not enough storage available (via the HESSIAN keyword) for the usual quasi-Newton method.

k	Conjugate gradient algorithm
1	Fletcher and Reeves (1964)
2	Polak and Ribiere (1969)
3	Perry (1976)
4	Memoryless DFP
5	Memoryless Complementary DFP

Further discussion is given in Section II.2.

● CRASH OPTION k

This determines which columns of [A I] are to be considered for construction of an initial basis B.

k	Meaning
0	The all-slack basis B = I is set up.
1	All columns of A are considered (except those corresponding to nonlinear variables which have been initialized in the MPS data at values away from their bounds -- see Section III.2). This is the default option.
≥ 2	The columns of A corresponding to <u>linear</u> variables will be considered (same as k = 1 if the problem is purely linear).

In all cases the initial basis is chosen to be strictly triangular without regard to feasibility or optimality.

1. The CRASH option is ignored if a starting basis is specified.

● DEBUG LEVEL k

Various positive values of k cause various amounts of additional information to be printed.

● DUMP FILE k

If k > 0, the last solution obtained will be output to file k in the format described in Section V.3. Variables are listed by state, name and value.

● ELEMENTS 3000

An over-estimate of the number of nonzero elements (coefficients, a_{ij}) in the constraint matrix.

1. COEFFICIENTS is a valid alternative keyword.
2. The default value is 5n where n is specified by the COLUMNS card.

● ERROR MESSAGE LIMIT 50

The maximum number of error messages to be printed for each type of error occurring during input. (Should be large for early runs on a particular MPS deck. Can be lowered to suppress warning of non-fatal errors during multiple subsequent runs.)

● FACTORIZATION FREQUENCY k

At most k iterations will occur between factorizations of the basis.

1. INVERT is a valid alternative keyword.
2. Normally k should be around 50 or 60. Values larger than 100 should almost never be used.

● HESSIAN DIMENSION k

This specifies that a $k \times k$ upper-triangular matrix R is to be available for use by the quasi-Newton algorithm, as a means of approximating the reduced Hessian matrix. Suppose that there are ns superbasic variables at a particular iteration.

1. If $ns \leq k$, the first ns columns of R will be used, although a constant $k(k + 1)/2$ words of storage are allocated from the beginning.
2. The storage for R is substantial if k is as large as 100 or 200. In general, k should be a slight overestimate of the final number of superbasic variables. If the number of nonlinear variables in the objective is nn then k need not be larger than $nn + 1$.
3. If ns exceeds k the quasi-Newton algorithm is discontinued in favor of a conjugate-gradient algorithm. The matrix R is not used until ns decreases to the value k.
4. The default value for k is n, where n is specified by a SUPERBASICS card, or 0 if the problem is linear.

● **IMBED**	YES
IMBED	NO

Determines whether part of the LU factorization of the basis is to be imbedded in the constraint matrix A.

1. If YES (the default), any non-spike column in B will be referenced in the L file by a single pointer to the appropriate column of A.
2. If NO, any non-spike column will be copied into the L file. This will increase the storage required for L but in general will reduce paging activity on a machine with virtual memory. Execution speed may therefore increase.

● INPUT FILE k

The file number for the MPS data file. The default value $(k = 5)$ refers to the card input stream.

● INSERT FILE k

References a file containing basis information in the format described in Section V.2.

1. Uses data produced on a PUNCH file.
2. Will be ignored if a positive OLD BASIS FILE is specified.

● INVERT FREQUENCY k

See FACTORIZATION FREQUENCY

● ITERATIONS k

An upper limit on the number of iterations allowed.

1. ITNS is a valid alternative keyword.
2. $k = 0$ is valid; feasibility is checked but optimality is not.
3. The default value is $3m + 10nn$ where m and nn are specified by ROWS and NONLINEAR VARIABLES respectively.

● LINESEARCH TOLERANCE t

For nonlinear problems, this controls the accuracy with which a minimum of the objective function will be located along the direction of search each iteration.

1. t must be a real number in the range $0.0 \leq t < 1.0$.
2. Smaller values of t generally result in more evaluations of the objective function but fewer total iterations.

3. The default value $t = 0.01$ requests a fairly accurate search, on the assumption that the total work per iteration is large relative to the work required in evaluating the objective function.
4. If the objective is expensive to evaluate, t should be in the range 0.1 to 0.9 (say).
5. The linesearch procedure[†] uses t in the following way. If d is the current search direction and $g(x + \alpha d)$ is the objective gradient at a step α along direction d, then the point $x + \alpha d$ will be accepted as an improvement over the point x if

$$\left| \frac{g(x + \alpha d)^T d}{g(x)^T d} \right| \le t$$

● LIST LIMIT 100

The maximum number of lines of MPS data to be listed during input. (The header cards, i.e., NAME, ROWS, COLUMNS, RHS, RANGE, BOUNDS, ENDATA, will always be listed along with their position in the MPS deck.)

● LOAD FILE k

References a file containing basis information in the format described in Section V.3.

1. Uses data produced on a DUMP file.
2. Will be ignored if a positive OLD BASIS FILE or INSERT FILE is specified.

[†]MINOS uses subroutines LNSRCH and NEWPTC from the NPL Algorithms Library. See Gill, Murray, Picken, Barber and Wright (1976).

• LOG FREQUENCY k

One line of the iteration log will be printed every k-th iteration. k = 10 or 20 is suggested for those interested only in the final solution.

• LOWER BOUND b

The default lower bound to be set on all structural variables before any bound set is input from the MPS file.

1. May be useful for bounding all variables away from singularities in the objective. (Other explicit bounds may also be necessary.)
2. If all or most variables are to be FREE, the negative value b = -1.0E+30 (or less) may be used to specify "minus infinity."

• LU ROW TOLERANCE t
  LU COL TOLERANCE t
  LU MOD TOLERANCE t

Tolerances for computing and updating LU factors of the basis. See Section VIII.

• MAXIMIZE
  MINIMIZE

The required direction of optimization (MIN is default). This applies to both linear and nonlinear terms in the objective.

• MULTIPLE PRICE k

If k > 0, this over-rides the PARTIAL PRICE parameter.

1. Whenever a PRICE operation is performed, the k best nonbasic variables will be selected for admission to the superbasic set.

2. Warning: If $k \geq 1$, this puts MINOS into "reduced gradient" mode even on purely linear problems. The subsequent iterations do not correspond to the very efficient "Minor Iterations" carried out by standard LP systems using multiple pricing. (All superbasics will be varied simultaneously. Note however that total storage requirements are independent of k; hence k need not be limited to 5 or 6 as it is in standard LP systems, which require k vectors of dimension m.)

3. On very large nonlinear problems it may be important to use quite a large value of k for the first few hundred or thousand iterations, as a means of "getting a lot of variables going" quickly. For example, if a problem has 3000 variables and 500 of them are nonlinear, the optimal solution may well have 200 or more variables superbasic. If the problem is solved in several runs, it may be beneficial to use $k = 10$ or 20 (say) for early runs, until it seems that the number of superbasics has levelled off.

4. The default strategy (no partial or multiple pricing, add one variable at a time to the superbasics) appears in practice to be preferable once a near-optimal solution has been obtained. This is partly because the projected Hessian approximation for s superbasics is likely to be improved more quickly for an expanded set of $s + k$ superbasics if k is only 1.

5. MULTIPLE PRICE 1 has a slightly different effect from PARTIAL PRICE 1, MULTIPLE PRICE 0, even though just 1 superbasic is added at each PRICE operation. For linear problems, the reduced-gradient mode of operation is slightly less efficient than the simplex algorithm. For nonlinear problems, the method of updating the reduced Hessian approximation differs. With MULTIPLE PRICE 1, the triangular matrix R is expanded to dimension $s + 1$ by appending the unit vector e_{s+1}. (Similarly if $k > 1$, k unit vectors are added.) Otherwise an attempt is made to obtain a better approximation, using some finite-difference technique.

● *NEW BASIS FILE* k

If $k > 0$, a bit map will be saved on file k every j-th iteration, where j is the specified SAVE FREQUENCY.

1. The first card of the file will contain the word PROCEEDING if the current run is still in progress (see Section V.1).
2. A bit map will also be saved at the end of a run (if $k > 0$).
3. This is the only basis file affected by the SAVE FREQUENCY parameter.

● NONLINEAR VARIABLES N

This specifies that the first N variables in the constraint matrix are to be regarded as "nonlinear" and made available as the array X(N) to subroutine CALCFG.

1. If $N = 0$, the problem is a linear program.

2. Not all N variables need be explicitly involved in the nonlinear objective. For example, some of them may be FIXED at permanent values. Note however that gradient values $G(j)$ must be supplied for all $j = 1, \ldots, N$. The correct value is 0.0 if variable j is essentially a linear variable, or c_j if the linear term $c_j x_j$ is not already included in the linear objective row.

3. In some cases it is convenient to regard even the logical variables as nonlinear (e.g., in solving the linear complementarity problem: $\min z^T w$ subject to $w = Mz + q$, $z^T w = 0$, $z \geq 0$, $w \geq 0$. This problem should be entered as a linear program of the form $Mz \geq -q$, $z \geq 0$, so that the "slack variables" may serve as $-w$.) The value of N should then be $n + 1 + m$ (allowing 1 for the RHS).

● OBJECTIVE COST

The 8-character name of the free row to be used as the linear part of the objective function.

1. If no OBJECTIVE card exists, or if the name specified is blank, the first free row will be selected.

2. If there are no free rows, or if the user (perhaps intentionally) specifies a non-existent OBJECTIVE name, optimization will terminate at the first feasible solution (unless a nonlinear objective exists).

3. If a nonlinear objective function is of the form $f(x_N) + c^Tx$, where $x = (x_N, x_L)$ is partitioned into nonlinear and linear variables respectively, the elements of c may be specified in several alternative ways. Let $c^Tx = c_N^Tx_N + c_L^Tx_L$. Then

   (a) all of c may be specified in the OBJ row of the MPS deck;

   (b) if $c_N^Tx_N$ has been programmed in subroutine CALCFG as part of the function $f(x_N)$, then only c_L should be specified in the MPS file;

   (c) either c_N or c_L or both may be absent.

• OLD BASIS FILE   k

Specifies a bit map to be used for the starting solution (see Section V.1).

1. $k > 0$ takes precedence over an INSERT file or a LOAD file.

• PARTIAL PRICE   k

The matrix A will be divided into k equal partitions $A_0, \ldots, A_{k-1}$ for the purposes of selecting a column to enter the basis. Normally one partition A_j will be priced, along with all of the slacks.

1. Almost always k should be set to 1, unless n is large and $m \ll n$, where A is m by n. The default value is $k = 1$.
2. Pricing stops if a variable in A_j or a slack variable has a favorable reduced gradient (as judged by a dynamic tolerance).
3. The next pricing starts with partition $A_{j+1 (\mathrm{mod}\ k)}$.

● PROBLEM NO. 3

For nonlinear problems, this allows the user to set the parameter "NPROB" in subroutine CALCFG. See Section III.3.

● PUNCH FILE k

If $k > 0$, the final solution obtained will be output to file k in the format described in Section V.2. For linear problems, this format is compatible with various commercial systems.

● RANGES RANGE001

The 8-character name of the range set to be selected from the MPS file.

1. RNG (or RNGS) are valid alternative keywords.
2. For multiple RANGE sets, see notes 2 and 3 under keyword BOUNDS and interpret appropriately.

● REDUCED GRADIENT TOLERANCE t

For nonlinear problems, this controls the extent to which optimization is restricted to the current set of basic and superbasic variables (i.e., Phase 4 iterations), before a new nonbasic variable is added to the superbasic set (Phase 3).

1. t must be a real number in the range $0.0 < t \leq 1.0$. The default value is $t = 0.2$.
2. Smaller values of t generally result in more iterations. Larger values may be satisfactory, particularly when the current solution is far from optimal.

3. Note: This parameter does not refer to the absolute size of the reduced gradient at any stage. It is not a means for requesting iterations to terminate when the reduced gradient reaches a certain value.

4. t is used in the following way. Whenever a variable x_j is added to the superbasic set the norm of the reduced-gradient vector (for all superbasics) is recorded. Let this be $\| \hat{g}_0 \|$. Usually $\| \hat{g}_0 \| = |d_j|$, where d_j is the "reduced cost" for the new superbasic. Subsequent Phase 4 iterations will continue at least until the norm of the reduced-gradient vector satisfies $\| \hat{g} \| \leq t \, \| \hat{g}_0 \|$. (Other convergence tests may cause further Phase 4 iterations.)

• RHS RHSIDE1

The 8-character name of the rhs vector to be selected from the MPS data.

1. See notes 2 and 3 under BOUNDS.
2. If no rhs exists after input (perhaps after deletion of small coefficients), a dummy zero vector is constructed.

• ROWS 500 .

An over-estimate of the number of rows in the constraint matrix.

1. If m is the number specified, the size of the hash table used to hold the row names will usually be $2m$. The number of collisions (and hence total input time) may in general be reduced by specifying m to be larger than necessary (within reason).

● SAVE FREQUENCY k

Suppose the NEW BASIS FILE card specified a file number f. If $f > 0$ and $k > 0$, a bit map describing the current solution will be saved on file f every k-th iteration.

● SOLUTION w

Determines whether a solution is to be printed on file 6 and/or output to a specified file, as follows:

<u>w</u>	<u>Meaning</u>
YES	The last solution obtained will be printed on file 6 (whether optimal or not).
NO	The solution will not be printed.
IF OPTIMAL	The solution will be printed only if it has been determined to be optimal, or if the problem appears to be infeasible or unbounded.
FILE k	If $k > 0$ the last solution obtained will be output to file k (whether optimal or not).

1. If w is blank or is anything other than YES, NO or FILE it will be treated as IF OPTIMAL.
2. The first 3 options (YES, NO, IF OPTIMAL) may co-exist with the FILE option.
3. For the 1st and 3rd options, floating-point numbers are printed with F16.5 format, and "infinite" upper or lower bounds are denoted by the word NONE.

4. For the FILE option, the format used is 1PE16.6 for all numbers, including "infinite" bounds which will appear as ±1.000000E+30.

5. Since the printer is file 6, the specifications

```
SOLUTION          NO
SOLUTION FILE      6
```

may be used together to obtain E-format in the printed solution (e.g., if 7 significant digits are required for all numbers).

● SUPERBASICS ns

Specifies how many superbasic variables are to be allowed for. Should be used in conjunction with a "HESSIAN DIMENSION" card.

1. In some sense, ns measures the expected deviation from linearity.
2. Normally ns need not be greater than nn + 1 where nn is the specified number of nonlinear variables.
3. For certain problems, ns may be considerably smaller than nn. This is important if nn is very large.

● TARGET OBJECTIVE VALUE t

This keyword invokes a certain optimization technique, and also sets the value of t. In effect it forces the solution at each iteration to be regarded as infeasible if the linear objective row has not yet reached the value t. This is designed to take advantage of the special pivot-row selection procedure (CHUZR) due to D.C. Rarick (1975), which minimizes the sum of infeasibilities at each iteration without concerning itself with the number of infeasibilities.

1. TARGET has no effect if there is no linear objective.
2. The value of t should take into account the direction of optimization (MIN or MAX).
3. It is generally unwise to invoke targeting unless t is known a priori to be sub-optimal.
4. If the problem appears to be infeasible, targeting is disabled and optimization continues normally.
5. Stay clear unless you know what you are doing.

● UPPER BOUND b

The default upper bound to be set on all structural variables before any bound set is input from the MPS file.

1. A value of 1.0E+30 or greater will be interpreted as "plus infinity." This is the default value and therefore need not be set.

● VERIFY w

Determines whether or not the computed gradient vector from the user's subroutine CALCFG is to be checked numerically. If w is the word NO, the gradient will not be checked. Any other value of w is equivalent to YES. Checking will be performed at the first feasible point found. Finite differences are taken of the objective value along two orthogonal directions. (If necessary, the directions will be modified to avoid stepping outside the feasible region.) The default value of w is NO.

• WEIGHT ON OBJECTIVE w

This keyword invokes the so-called composite objective technique, if the first solution obtained is infeasible. The method attempts to optimize the true objective function (linear terms only) while simulteneously reducing the sum of infeasibilities.

1. The objective function is defined at each infeasible iteration to be

$$\min mw(c^T x) + (\text{sum of infeasibilities})$$

   where $m = 1$ for MIN, -1 for MAX, and c is the linear objective row.

2. If the sum of infeasibilities can not be reduced at some stage, w is reduced by a factor of 10. This helps to allow for the possibility that the w specified by the user is too large. It also provides dynamic allowance for the fact that the sum of infeasibilities is tending towards zero.

3. The effect of w is disabled after 5 such reductions of w, or if a feasible solution is obtained.

V. Basis File Formats

Figures 1 through 4 illustrate the data formats used for the various basis files introduced in Section III.4. Selected column numbers are included to define the significant data fields. The logical records in each file are alphanumeric data with the following record lengths:

File	Logical record length
OLD/NEW BIT MAP	80
PUNCH/INSERT FILE	36
DUMP/LOAD FILE	36
SOLUTION FILE	108

(Larger record lengths may be used for convenience, e.g., 80, 80, 80, 133 respectively.)

The files shown correspond to the optimal solution for a small but non-trivial example, namely the Weapon Assignment problem in Himmelblau (1972), problem 23. This is of the form

$$\text{minimize} \quad \sum_{j=1}^{20} u_j \Big(\prod_{i=1}^{5} a_{ij}^{x_{ij}} - 1 \Big)$$

$$\text{subject to} \quad Ax \overset{\geq}{\leq} b, \qquad x \geq 0$$

where A has dimensions 12×100.

V.1. OLD/NEW Basis Files (bit maps)

A bit map is primarily intended for restarting the solution of a problem at a point that was reached by an earlier run, either on the same problem or on a related problem of the same dimensions, e.g., one which has a different OBJECTIVE, RHS, RANGE or BOUND set or different matrix coefficients.

```
1      8      15      23    29          40  43     50      57 59                  80
:      :      :       :     :           :   :      :       :  :                   :
WEAPON A      ITN     122   OPTIMAL SOLN    PHASE  3       OBJ   -1.7355695798560+03
OBJ=          RHS=B         RNG=            BND=           M=    12  N=   113  SB= 18
00003220020000202000020003000030000200002000000003000300002000002000300330000203 0
0220000200002000030010000003122123
      41     2.0316768382961 90+01
      82     3.8027533814957 7D+00
      17     2.3489533295527 3D+01
      36     2.7066733129639 2D+01
      59     4.0934258239765 5D+01
      22     2.0899605797884 0D+01
      15     4.8628858754640 8D+01
      88     5.7551539944994 5D+01
      65     5.4015197600566 4D+01
     112    -6.2810931659367 3D+00
      93     6.4211504061432 7D+01
     109    -1.1132295900964 5D+01
       6     1.3516456395590 1D+01
      77     2.4236467215620 1D+01
     110    -8.8237529151783 9D+00
      83     7.2033835250498 1D+01
       7     1.3604576427287 7D+00
      10     4.5408197027647 6D+01
       0
:      :                        :                                                  :
1      8                        32                                                 80
```

Figure 1.   Format of OLD and NEW BASIS FILEs (Bit Maps)

The information saved is very compact (but rather difficult to modify). With some computer systems it is possible to access the bit map at occasional intervals during a run; in such cases the first card of the file will provide a useful glimpse of the run's progress so far.

As illustrated in Figure 1, the information contained in an *OLD* or *NEW BASIS FILE* is as follows:

1. A card containing the problem name, iteration number, current status (OPTIMAL SOLN, INFEASIBLE, UNBOUNDED, EXCESS ITNS, ERROR CONDN, or PROCEEDING), current phase and current objective value or sum of infeasibilities.
2. A card containing the OBJ, RHS, RNG and BND names, M = no. of rows in the constraint matrix, N = total no. of variables, SB = current no. of superbasics. In this context,

$$N = n + 1 + m$$

   where

   (a) the first n variables are those in A;

   (b) variable $n + 1$ is the RHS (fixed at -1.0);

   (c) the last m variables are the logicals (slacks).

3. A set of (N-1)/80 + 1 cards indicating the state of each of the above N variables. One character HS(j) is recorded for each $j = 1, 2, \ldots, N$ as follows, written with FORMAT(80I1):

HS(j)	State of the j-th variable
0	Nonbasic at lower bound
1	Nonbasic at upper bound
2	Superbasic
3	Basic

If variable j is FIXED (lower bound = upper bound), then HS(j) may be 0 or 1. The same is true if variable j is FREE (infinite bounds) and still nonbasic, although free variables will usually be basic.

4. A set of cards of the form

$$j \qquad x_j$$

where j is a variable number and x_j is a value. These are written with FORMAT(I8, 1PE24.14) and are terminated by a card with $j = 0$. Normally j will be in the range 1 to N and will correspond to a variable for which HS(j) = 2 (superbasic).

Notes on Input of an OLD BASIS FILE

1. Card 1 above is printed but not otherwise used.
2. The values labelled M and N on card 2 must agree with those for the MPS file that has just been read. The value labelled SB is read and printed as an integer but is not otherwise used.
3. Cards 3, 4, etc., must contain exactly M values HS(j) = 3 (for the basic variables).

4. The cards containing j and x_j values need not occur in natural column order (e.g., see Figure 1). The user may modify or extend this list of cards relatively easily as long as the following points are borne in mind.

5. The list should include a card for all variables whose state is $HS(j) = 2$ (superbasic).

6. For any given j and x_j, if $HS(j) \neq 3$ then variable j will normally be initialized at the value x_j and its state will be reset to $HS(j) = 2$ (superbasic).

7. If $HS(j) = 3$, the value x_j will be recorded for nonlinear variables but no other action will be taken (thus variable j will remain basic).

8. If the specified value x_j lies on or outside the bounds on variable j, or if the total number of superbasics has already reached the limit specified in the SPECS file, then variable j will be made nonbasic at the bound nearest to x_j.

V.2. PUNCH and INSERT files (industry standard)

These files specify a basis by a list of cards of the form

INDICATOR NAME1 NAME2 VALUE

as shown in Figure 2. The output to a PUNCH file may be used directly as input from an INSERT file, either for MINOS or (if the problem is purely linear) for one of the commercial LP systems. The various indicators are best defined in terms of the action taken on input. It is assumed that the basis is initially the set of logical variables and that the structurals are nonbasic at their smallest bound in absolute magnitude.

```
NAME          WEAPON A  PUNCH/INSERT
 XL X015      L1        5.081550D+01
 SB X021                1.351650D+01
 SB X022                1.360460D+00
 SB X025                4.540820D+01
 SB X035                4.862890D+01
 SB X042                2.348950D+01
 SB X052                2.089960D+01
 XL X061      L2        1.000000D+02
 XL X071      L3        3.910000D+01
 SB X081                2.706670D+01
 SB X091                2.031680D+01
 XL X105      L4        5.113230D+01
 XL X114      L5        3.319740D+01
 SB X124                4.093430D+01
 SB X135                5.401520D+01
 XU X144      G7        5.882380D+01
 XL X152      G8        2.621120D+01
 XL X153      G9        4.378880D+01
 SB X162                2.423650D+01
 XU X164      G10       1.704460D+01
 SB X172                3.802750D+00
 SB X173                7.203380D+01
 SB X183                5.755150D+01
 SB X193                6.421150D+01
 XL X203      G11       6.241430D+01
 SB G8                 -1.113230D+01
 SB G9                 -8.823750D+00
 SB G11                -6.281090D+00
ENDATA

|   |      |  |      |  |          |
1   5     12  15    22  25         36
```

Figure 2.   Format of PUNCH and INSERT FILEs

<u>INDICATOR</u>	<u>Action Taken During INSERT</u>
XL (XU)	Make structural NAME1 basic and logical NAME2 nonbasic at its lower (upper) bound.
LL (UL)	Make variable NAME1 nonbasic at its lower (upper) bound.
SB	Make variable NAME1 superbasic at the value VALUE.

<u>Notes on PUNCH Data</u>

1. Variables are output in natural order. For example, on the first XL or XU card, NAME1 will be the first basic structural and NAME2 will be the first logical which is not basic (it may be nonbasic or superbasic).
2. Indicator SB is an addition to the standard format to allow for non-vertex solutions.
3. Superbasic logicals are output last. These will be some of the logicals which previously appeared as NAME2 (e.g., logicals G8, G9, G11 in Figure 2).
4. PUNCH and INSERT files deal with the status and value of <u>logical variables</u>, whereas SOLUTION files deal with <u>row</u> status and activities.
5. LL or UL cards are <u>not</u> output for nonbasic variables if the corresponding bound value is zero.

Notes on INSERT Data

1. Preferably an INSERT file should be an unmodified PUNCH file obtained from an earlier run on the same problem.
2. VALUE is used only if INDICATOR is SB or NAME1 is a nonlinear variable.
3. Warning -- if D or E format is used for a VALUE, the exponent must be right-justified to column 36.
4. Additional SB cards may be added before the ENDATA card, to specify additional superbasic structurals or logicals. (Any such card will be ignored if previous cards have implied that NAME1 is basic or superbasic, or that NAME2 is not basic.)
5. If the specified VALUE on an SB card lies on or outside the bounds on variable NAME1, or if the total number of superbasics has already reached the limit specified in the SPECS files, then variable NAME1 will be made nonbasic at the bound nearest to VALUE.

V.3. DUMP and LOAD Files

These files are similar to PUNCH and INSERT files but record solution information in a manner that is more direct and more easily modified. In particular, no distinction is made between row and column variables. (Note that status indicators and values refer to rows as in the SOLUTION file, rather than the corresponding logical variables.)

```
NAME          WEAPON A     DUMP/LOAD
 BS X015                 5.081550+01
 SB X021                 1.351650+01
 SB X022                 1.360460+00
 SB X025                 4.540820+01
 SB X035                 4.862890+01
 SB X042                 2.348950+01
 SB X052                 2.089960+01
 BS X061                 1.000000+02
 BS X071                 3.910000+01
 SB X081                 2.706670+01
 SB X091                 2.031680+01
 BS X105                 5.113230+01
 BS X114                 3.319740+01
 SB X124                 4.093430+01
 SB X135                 5.401520+01
 BS X144                 5.882380+01
 BS X152                 2.621120+01
 BS X153                 4.378880+01
 SB X162                 2.423650+01
 BS X164                 1.704460+01
 SB X172                 3.802750+00
 SB X173                 7.203380+01
 SB X183                 5.755150+01
 SB X193                 6.421150+01
 BS X203                 6.241430+01
 BS G6                  -2.081550+01
 SB G8                  -1.113230+01
 SB G9                  -8.823750+00
 SB G11                 -6.281090+00
 BS G12                 -5.241430+01
ENDATA

|   |      |           |          |
1   5      12          25         36
```

Figure 3.   Format of DUMP and LOAD FILEs

Each data card is of the form

INDICATOR   NAME   (blank)   VALUE

as shown in Figure 3. The indicators LL, UL, BS and SB mean Lower Limit, Upper Limit, Basic and Superbasic respectively.

Notes on DUMP Data

1. Variables are output in natural order, columns first, then rows.
2. LL or UL cards are not output for nonbasic variables if the corresponding bound value is zero, or if the variable NAME is FIXED (lower bound = upper bound).
3. Nonbasic FREE variables will be output with either LL or UL indicators but with value zero.

Notes on LOAD Data

1. Normally a DUMP file will be valid as a LOAD file to restart solution of the same problem.
2. VALUE is used only if INDICATOR is SB or if NAME is a nonlinear variable.
3. If the VALUE on an SB card lies on or outside the bounds on variable NAME, or if the total number of superbasics has already reached the limit specified in the SPECS file, then variable NAME will be made nonbasic at the bound nearest to VALUE.

4. (Partial basis) If fewer than M variables are specified to be basic, a tentative basis list will be constructed by adding the requisite number of logicals, starting from the first row and taking those that were not previously specified to be basic or superbasic.

   Note: If the resulting basis is singular, the basis factorization routine will reject a number of variables and introduce an appropriate set of logicals. The rejected variables will be favored during subsequent iterations for readmission to the basis, but otherwise the starting point obtained will not necessarily be "good."

5. (Too many basics) If M variables have already been specified as basic, any further BS indicators will be treated as though they were SB. This feature may be useful in combining solutions to smaller subproblems.

V.4. SOLUTION File

Figure 4 illustrates the format of the file output at the end of a run if requested by a SPECS card of the form

```
SOLUTION FILE     k
```

$(k > 0)$.

```
1          10         20         30         40         50         60         70         80         90         100        108

1
 PROBLEM NAME:      WEAPON A               OBJECTIVE VALUE:        -1.735569579ᴅD+03

 STATUS:            OPTIMAL SOLN           PHASE:           3       ITERATION:     122

 OBJECTIVE:                  (MIN)
 RHS:               B
 RANGES:
 BOUNDS:

 SECTION 1 - ROWS

  NUMBER   ...ROW..   AT   ...ACTIVITY...   SLACK ACTIVITY   ..LOWER LIMIT.   ..UPPER LIMIT.   .DUAL ACTIVITY    ..I

     102   L1         UL   2.0000000D+02     0.0             -1.000000E+30     2.000000E+02    -5.992750D-02      1
     103   L2         UL   1.0000000D+02     0.0             -1.000000E+30     1.000000E+02    -2.176944D-01      2
     104   L3         UL   3.0000000D+02     0.0             -1.000000E+30     3.000000E+02    -6.870748D-02      3
     105   L4         UL   1.5000000D+02     0.0             -1.000000E+30     1.500000E+02    -1.235846D-01      4
     106   L5         UL   2.5000000D+02     0.0             -1.000000E+30     2.500000E+02    -7.229390D-02      5
     107   G6         BS   5.0815450D+01    -2.0815450D+01     3.000000E+01     1.000000E+30     0.0               6
     108   G7         LL   1.0000000D+02     0.0              1.000000E+02     1.000000E+30     5.992564D-02      7
     109   G8        SBS   5.1132300D+01    -1.1132300D+01     4.000000E+01     1.000000E+30     7.836420D-09      8
     110   G9        SBS   5.8623750D+01    -8.6237530D+00     5.000000E+01     1.000000E+30    -1.209050D-09      9
     111   G10        LL   7.0000000D+01     0.0              7.000000E+01     1.000000E+30     2.699879D-02     10
     112   G11       SBS   4.1281090D+01    -6.2810930D+00     3.500000E+01     1.000000E+30    -1.884039D-09     11
     113   G12        BS   6.2414300D+01    -5.2414300D+01     1.000000E+01     1.000000E+30     0.0              12

1      8 10      18  22              38               54               70               86               102    108

       I5  2X,2A4    A4          E16.6            E16.6            E16.6            E16.6            E16.6     I6
```

Figure 4. Format of the SOLUTION File

```
1
SECTION 2 - COLUMNS

NUMBER  .COLUMN.  AT    ...ACTIVITY...    .OBJ GRADIENT.   ..LOWER LIMIT.   ..UPPER LIMIT.   .REDUCED COST.   M+J

    1   X011      LL    0.0                0.0              0.0              1.000000E+30     5.992750D-02     13
    2   X012      LL    0.0               -1.511624D-01     0.0              1.000000E+30     6.553205D-02     14
    3   X013      LL    0.0               -3.539220D-02     0.0              1.000000E+30     3.331527D-02     15
    4   X014      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     16
    5   X015      BS    5.081545D+01      -7.229090D-02     0.0              1.000000E+30     0.0              17
    6   X021     SBS    1.351646D+01      -5.992750D-02     0.0              1.000000E+30     0.0              18
    7   X022     SBS    1.360458D+00      -2.175944D-01     0.0              1.000000E+30     0.0              19
    8   X023      LL    0.0               -5.992750D-02     0.0              1.000000E+30     8.779980D-03     20
    9   X024      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     21
   10   X025     SBS    4.540820D+01      -7.229089D-02     0.0              1.000000E+30     0.0              22
   11   X031      LL    0.0                0.0              0.0              1.000000E+30     5.992750D-02     23
   12   X032      LL    0.0               -1.409021D-01     0.0              1.000000E+30     7.579239D-02     24
   13   X033      LL    0.0               -3.539220D-02     0.0              1.000000E+30     3.331527D-02     25
   14   X034      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     26
   15   X035     SBS    4.862886D+01      -7.229090D-02     0.0              1.000000E+30     0.0              27
   16   X041      LL    0.0                0.0              0.0              1.000000E+30     5.992750D-02     28
   17   X042     SBS    2.348955D+01      -2.176945D-01     0.0              1.000000E+30     0.0              29
   18   X043      LL    0.0               -5.096963D-02     0.0              1.000000E+30     1.773787D-02     30
   19   X044      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     31
   20   X045      LL    0.0               -6.404388D-02     0.0              1.000000E+30     8.247024D-03     32
   21   X051      LL    0.0                0.0              0.0              1.000000E+30     5.992750D-02     33
   22   X052     SBS    2.089961D+01      -2.176944D-01     0.0              1.000000E+30     0.0              34
   23   X053      LL    0.0               -5.468114D-02     0.0              1.000000E+30     1.402633D-02     35
   24   X054      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     36
   25   X055      LL    0.0               -6.870747D-02     0.0              1.000000E+30     3.583433D-03     37
   26   X061      BS    1.000000D+02      -8.529975D-07     0.0              1.000000E+30     0.0              38
   27   X062      LL    0.0               -1.105991D-06     0.0              1.000000E+30     1.577567D-01     39
   28   X063      LL    0.0               -5.529955D-07     0.0              1.000000E+30     8.780282D-03     40
   29   X064      LL    0.0                0.0              0.0              1.000000E+30     6.355795D-02     41
   30   X065      LL    0.0               -1.060360D-07     0.0              1.000000E+30     1.236415D-02     42
   31   X071      BS    3.910004D+01      -5.992750D-02     0.0              1.000000E+30     0.0              43
   32   X072      LL    0.0               -1.198550D-01     0.0              1.000000E+30     9.783945D-02     44
   33   X073      LL    0.0               -4.742622D-02     0.0              1.000000E+30     2.128126D-02     45
   34   X074      LL    0.0                0.0              0.0              1.000000E+30     1.235846D-01     46
   35   X075      LL    0.0               -1.149100D-02     0.0              1.000000E+30     6.079990D-02     47
   36   X081     SBS    2.706073D+01      -5.992749D-02     0.0              1.000000E+30     0.0              48
   37   X082      LL    0.0               -7.317712D-02     0.0              1.000000E+30     1.445173D-01     49
   38   X083      LL    0.0               -3.477627D-02     0.0              1.000000E+30     3.393120D-02     50
   39   X084      LL    0.0                0.0              0.0              1.000000E+30     1.235845D-01     51
   40   X085      LL    0.0                0.0              0.0              1.000000E+30     7.229090D-02     52
   41   X091     SBS    2.031677D+01      -5.992749D-02     0.0              1.000000E+30     0.0              53
   42   X092      LL    0.0               -5.992749D-02     0.0              1.000000E+30     1.577670D-01     54
   43   X093      LL    0.0               -2.239299D-02     0.0              1.000000E+30     4.531449D-02     55
   44   X094      LL    0.0                0.0              0.0              1.000000E+30     1.235845D-01     56
   45   X095      LL    0.0                0.0              0.0              1.000000E+30     7.229090D-02     57
   46   X101      LL    0.0                0.0              0.0              1.000000E+30     5.992749D-02     58
   47   X102      LL    0.0               -1.034839D-01     0.0              1.000000E+30     1.142105D-01     59
   48   X103      LL    0.0               -3.519381D-02     0.0              1.000000E+30     3.351366D-02     60
   49   X104      LL    0.0               -2.800915D-02     0.0              1.000000E+30     9.557544D-02     61
   50   X105      BS    5.113233D+01      -7.229089D-02     0.0              1.000000E+30     0.0              62
   51   X111      --    0.0                0.0              0.0              1.000000E+30     5.992750D-02     63
   52   X112      LL    0.0                0.0              0.0              1.000000E+30     2.176944D-01     64
   53   X113      LL    0.0               -1.316995D-02     0.0              1.000000E+30     5.553753D-02     65
   54   X114      BS    3.319736D+01      -1.235845D-01     0.0              1.000000E+30     0.0              66
   55   X115      LL    0.0               -8.721467D-02     0.0              1.000000E+30     5.076230D-03     67
```

```
 56  X121   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02   68
 57  X122   LL   0.0            -2.994358D-02    0.0             1.000000E+30   1.377509D-01   69
 58  X123   LL   0.0            -2.994358D-02    0.0             1.000000E+30   3.876390D-02   70
 59  X124   SBS  4.093426D+01   -1.235846D-01    0.0             1.000000E+30   0.0            71
 60  X125   LL   0.0            -6.050459D-02    0.0             1.000000E+30   1.178631D-02   72
 61  X131   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02   73
 62  X132   LL   0.0             0.0             0.0             1.000000E+30   2.175944D-01   74
 63  X133   LL   0.0            -7.703771D-03    0.0             1.000000E+30   6.100371D-02   75
 64  X134   LL   0.0            -7.229089D-02    0.0             1.000000E+30   5.129371D-02   76
 65  X135   SBS  5.401520D+01   -7.229089D-02    0.0             1.000000E+30   0.0            77
 66  X141   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02   78
 67  X142   LL   0.0            -1.894691D-01    0.0             1.000000E+30   2.822537D-02   79
 68  X143   LL   0.0            -2.994358D-02    0.0             1.000000E+30   3.876390D-02   80
 69  X144   BS   5.882375D+01   -1.235846D-01    0.0             1.000000E+30   0.0            81
 70  X145   LL   0.0            -2.994358D-02    0.0             1.000000E+30   4.234733D-02   82
 71  X151   LL   0.0             0.0             0.0             1.000000E+30   3.292871D-02   83
 72  X152   BS   2.621118D+01   -1.906957D-01    0.0             1.000000E+30   0.0            84
 73  X153   BS   4.378882D+01   -4.170859D-02    0.0             1.000000E+30   0.0            85
 74  X154   LL   0.0            -2.706415D-02    0.0             1.000000E+30   6.392165D-02   86
 75  X155   LL   0.0            -1.376222D-02    0.0             1.000000E+30   3.152990D-02   87
 76  X161   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02   88
 77  X162   SBS  2.423547D+01   -2.176944D-01    0.0             1.000000E+30   0.0            89
 78  X163   LL   0.0            -3.440430D-02    0.0             1.000000E+30   3.430318D-02   90
 79  X164   BS   1.704463D+01   -1.235846D-01    0.0             1.000000E+30   0.0            91
 80  X165   LL   0.0            -1.711527D-02    0.0             1.000000E+30   5.517564D-02   92
 81  X171   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02   93
 82  X172   SBS  3.602753D+00   -2.176945D-01    0.0             1.000000E+30   0.0            94
 83  X173   SBS  7.203384D+01   -6.870745D-02    0.0             1.000000E+30   0.0            95
 84  X174   LL   0.0             0.0             0.0             1.000000E+30   1.235845D-01   96
 85  X175   LL   0.0             0.0             0.0             1.000000E+30   7.229090D-02   97
 86  X181   LL   0.0            -4.226631D-02    0.0             1.000000E+30   1.766119D-02   98
 87  X182   LL   0.0            -1.436693D-01    0.0             1.000000E+30   7.402511D-02   99
 88  X183   SBS  5.755154D+01   -6.870747D-02    0.0             1.000000E+30   0.0           100
 89  X184   LL   0.0             0.0             0.0             1.000000E+30   1.235846D-01  101
 90  X185   LL   0.0             0.0             0.0             1.000000E+30   7.229090D-02  102
 91  X191   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02  103
 92  X192   LL   0.0            -1.538674D-01    0.0             1.000000E+30   6.382702D-02  104
 93  X193   SBS  6.421150D+01   -6.870743D-02    0.0             1.000000E+30   0.0           105
 94  X194   LL   0.0             0.0             0.0             1.000000E+30   1.235846D-01  106
 95  X195   LL   0.0             0.0             0.0             1.000000E+30   7.229090D-02  107
 96  X201   LL   0.0             0.0             0.0             1.000000E+30   5.992750D-02  108
 97  X202   LL   0.0            -1.339176D-01    0.0             1.000000E+30   8.377684D-02  109
 98  X203   BS   6.241430D+01   -6.870748D-02    0.0             1.000000E+30   0.0           110
 99  X204   LL   0.0             0.0             0.0             1.000000E+30   1.235845D-01  111
100  X205   LL   0.0             0.0             0.0             1.000000E+30   7.229090D-02  112
101  B      EQ  -1.000000D+00    0.0            -1.000000E+00   -1.000000E+00   8.309502D+01  113
```

The format of a <u>printed</u> solution is essentially identical except for the following:

1. The first character in each line is for carriage control and will not appear on the printed page. Only two lines have a non-blank carriage control.
2. Format F16.5 is used for numerical values rather than 1PE16.6.
3. Any UPPER or LOWER LIMITS that are "infinite" ($\pm$ 1.0E+30) appear as the word NONE.

In the ROWS section, both ROW and SLACK activities are shown. The quantities labelled DUAL ACTIVITY are the *simplex multipliers*, π.

In the COLUMNS section, OBJ GRADIENT means, for linear variables, the coefficient in the objective row, say c_j. For nonlinear variables, if the solution is feasible, it means $g_j + c_j$ where g_j is the relevant component of the gradient of the nonlinear function $f(x)$. (If the solution is infeasible, just c_j appears.)

The quantities labelled REDUCED COST are the reduced gradients $d_j = c_j - \pi^T a_j$ (or $g_j + c_j - \pi^T a_j$) for each column a_j. At an optimal solution these should be zero for basic variables (status BS), non-negative for variables at LL, non-positive for variables at UL, and zero or very small (in absolute magnitude) for variables labelled SBS.

If the solution is infeasible, the status of basic and superbasic variables (rows or columns) may be shown as follows:

++	variable is above its upper limit
--	variable is below its lower limit.

A SOLUTION file is intended to be read from disk by a self-contained program which extracts and saves certain values as required, possibly for further computation. Typically the first 14 lines would be skipped. Each subsequent line should be read using a format of the form

```
FORMAT(I8, 2X, 2A4, A4, 5E16.6, I6)
```

The end of the ROWS section can be detected by the almost blank line. If this and the next 4 lines are skipped, the COLUMNS section can then be read under the same format. Hence the inefficiency of BACKSPACE or re-read operations can be avoided.

VI. Restarting Modified Problems

The preceding sections document three distinct starting methods (OLD BIT MAP, INSERT, LOAD) which may be preferable to any of the three cold start (CRASH) options. The best choice depends on the extent to which the present problem has been modified, and whether variables are to be specified by name or by number.

1. (Protection) In general the user is protected against accidental specification of infinite bounds. For example, if UL is specified for some variable that has an upper bound of plus infinity, the indicator is treated as if it were LL. Similarly for LL cards, if the variable's lower bound is minus infinity. No message is issued. (Note -- switching bounds is safe even with nonbasic FREE variables. They will be treated as having value zero.)

2. (Default status) If the status of a variable is not explicitly specified, it will initially be nonbasic at the bound which is <u>smallest in absolute magnitude</u>.

3. (Relaxing of bounds) Suppose that in Problem A a certain variable X has bounds

$$0 \leq X \leq 5.0$$

and that when a solution is saved, X is nonbasic at its upper bound 5.0. Certain care must be taken if the user wishes to use the solution to start some Problem B in which the upper bound on X has been relaxed. If <u>no</u> action is taken, two cases arise:

(a) If the upper bound has been <u>removed</u>, a bound switch would normally occur (e.g., UL would be treated as LL) and X would be made nonbasic at its <u>lower</u> bound, 0.0.

(b) If the bound has been <u>relaxed</u> to some larger but finite value, e.g., $0 \leq X \leq 10.0$, X would normally be made nonbasic at that upper bound.

In both cases the new solution will clearly be different from the old. In some situations this may not be of consequence, but often the new solution may prove to be severely infeasible or seriously removed from optimality.

The same comments apply if X were FIXED at 5.0 in Problem A.

To retain the original solution precisely (all other things being equal), the user need only request that X be made superbasic at the required value 5.0. This means inserting a card of the form

```
j     5.0
```

near the end of a bit map, or an

```
SB     X        5.0
```

card near the end of an INSERT or LOAD file.

In all cases, the necessary storage must be requested by a SPECS card of the form

```
SUPERBASICS        k
```

where k is at least as large as the number of superbasics being specified.

4. In case (a) above the user may happen to know that the upper bound 5.0 in Problem A will effectively be a lower bound on the optimal value of X in Problem B. Of course it would then be simple to define the bounds on X to be

$$5.0 \leq X \leq \infty$$

in the MPS data for Problem B, and no further action need be taken. In case (b), the bounds

$$5.0 \leq X \leq 10.0$$

could be included, and the state of X would have to be changed from UL to LL. However, when many interacting variables are involved it may be difficult to determine in advance that X will move above its old bound; the previous method (making X superbasic) is then much to be preferred.

5. (Tightening of bounds) If Problems A and B have the bounds

$$0 \leq X \leq 5.0$$

$$0 \leq X \leq 2.0$$

respectively, and if X is again at its upper limit in Problem A, there is no way in general of initializing X at value 5.0 for Problem B. MINOS will move X to its nearest bound, in this case 2.0, and hopefully most users will be happy. However, note the following rule of thumb.

6. (Other modifications) Wherever possible, a series of problems should be ordered so that the most tightly constrained cases are solved <u>first</u>. Their solutions will often provide feasible starting points for subsequent relaxed problems.

VII. The Iteration Log

One line of information is printed on file 6 every k-th iteration, where k is specified by a card of the form LOG FREQUENCY k. The default value is $k = 1$. For large problems, $k = 10$ will save a significant volume of paper.

In the following description, a PRICE operation is defined to be the process by which one or more nonbasic variables are selected to become superbasic. Normally just one variable is selected, which we will denote by J. If the problem is purely linear, variable J will usually become basic immediately (unless it should happen to reach its opposite bound and return to the nonbasic set).

If Partial Pricing is in effect, variable J is selected from the matrix $[A_{PP}\ I]$ where A_{PP} is the PP-th partition of A, and I corresponds to the slack variables.

If Multiple Pricing is in effect, several variables may be selected from the whole matrix $[A\ I]$. In this case J refers to the variable with the largest favorable "reduced cost."

The iteration log includes the following items.

ITN	The current iteration number.
PH	The current phase, as follows: 1 => Phase 1 simplex procedure (the current solution is infeasible). 2 => Phase 2 simplex procedure (e.g., if the problem is a linear program, or if none of the nonlinear variables is basic). 3 => A PRICE operation has just been performed and the current iteration is a non-simplex step. 4 => Optimization is being performed amongst the current basic and superbasic variables (ignoring the nonbasics).
PP	The "Partial Price" indicator, reset by the PRICE operation in Phase 1, 2, or 3. PP = 1 if partial pricing is not in effect. PP = 0 if variable J is one of those rejected from the basis by the previous basis factorization.
NOPT	The number of "non-optimal" variables present in whatever set of nonbasics was scanned during the last PRICE operation. Reset in Phase 1, 2, or 3.
DJ/RG	In Phase 1, 2, or 3 this is "DJ," the reduced cost of the variable J selected by PRICE. In Phase 4 it is "RG," the norm of the reduced gradient vector; i.e., the largest reduced gradient amongst the current set of superbasics.

+SBS	The variable J selected by PRICE to become superbasic.
-SBS	The variable chosen to leave the set of superbasics (to become either basic or nonbasic).
-BS	The variable chosen to leave the basis to become nonbasic.
STEP	The distance moved along the current search direction.
PIVOT	If column a_q replaces the p-th column of the basis B, this is the p-th element of the transformed column $B^{-1}a_q$.
NSPK	The current number of spikes in the LU factorization of B. This is the number of non-trivial columns in U (excluding ones that have been deleted in earlier iterations).
L	The number of nonzeros in the L part of the factorization of B (excluding elements that are imbedded in the constraint matrix A).
U	The number of nonzeros in the U part of the factorization of B (including ones in spikes that have been deleted in earlier iterations). This does not include nonzeros in a certain matrix F, which is really part of U but is stored separately as an upper triangular matrix of dimension NSPK.
NINF	The number of infeasibilities <u>before</u> the present iteration. (Note -- this number need not decrease monotonically.)

SINF/ OBJECTIVE	If NINF > 0, this is SINF, the sum of infeasibilities before the present iteration; it should decrease monotonically. If NINF $= 0$, the value of the true objective after the present iteration is given (including both linear and nonlinear terms).

The following items are printed if the problem is nonlinear or if the Phase is 3 or 4.

NFG	The number of times the nonlinear objective and its gradient have been evaluated.
NSB	The current number of superbasic variables.
R1M	An indication of the type of modifications that have been made to the matrix R being used to approximate the projected Hessian matrix.
H-CONDN	This is an estimate of the condition number of the reduced Hessian matrix. More precisely it is K, the square of the ratio of the largest and smallest diagonals of the triangular matrix R, which constitutes a lower bound on the condition number of the matrix $(R^T R)$ that approximates the projected Hessian. The value of K gives a rough indication of whether or not the optimization procedure is having difficulty. Let ϵ be the relative precision of the floating-point arithmetic being used. The reduced-gradient algorithm will make slow progress if K becomes as large as $\epsilon^{-1/2}$,

and will probably fail to find a better solution if K reaches $\epsilon^{-3/4}$ or larger. In such cases the user should

(1) check that the constraint matrix is well scaled;

(2) scale the variables involved in the nonlinear objective so that the Hessian matrix is as well conditioned as possible (ignoring rows and columns corresponding to nonbasic variables);

(3) add upper or lower bounds to certain variables to keep them a reasonable distance from any singularities in the objective or its derivatives.

CONV A set of four logical variables C_1, C_2, C_3, C_4 that are used to determine when to discontinue optimization in the current subspace (Phase 4) and consider moving off a constraint (Phase 3).

Let RG be the current norm of the reduced gradient. The meaning of the C_j is briefly as follows:

C_1 is TRUE if the change in x was sufficiently small.

C_2 is TRUE if the change in $f(x)$ was sufficiently small.

C_3 is TRUE if RG is smaller than some loose tolerance.

C_4 is TRUE if RG is smaller than some tight tolerance.

The test used is of the form <u>if</u> (C_1 <u>and</u> C_2 <u>and</u> C_3) <u>or</u> C_4 <u>then</u> <u>go</u> <u>to</u> Phase 3. All tolerances are varied dynamically.

VIII. Tolerances for Computing the Basis Factorization

Three tolerances T_r, T_c and T_m are involved in computing and updating the LU factorization of the basis. They may be set by specification cards of the following form (their default values are shown as comments):

```
LU ROW TOLERANCE = 0.01        * DEFAULT Tr = 0.001
LU COL TOLERANCE = 0.9         * DEFAULT Tc = 0.1
LU MOD TOLERANCE = 1.0         * DEFAULT Tm = 0.99
```

The use of T_r and T_c must be understood in terms of the "bump and spike" structure of a basis matrix as introduced by Hellerman and Rarick (1971, 1972) in their Preassigned Pivot Procedures P^3 and P^4. The general aim is to accept the preassigned pivot order whenever possible (since it has close to the minimal number of spikes) but to revise this order where necessary to maintain numerical stability.

In this context, attention is restricted to the rows and columns inside any one bump, and the diagonal elements between bumps are regarded as bumps of dimension 1. Each column in a bump is initially either a <u>spike</u> column or a <u>triangle</u> (non-spike)column.

The k-th column in a bump fails a <u>row</u> test if its preassigned pivot element is smaller than T_r times the largest remaining potential pivot in that column. Some replacement column is then selected from the remaining spikes in the bump as follows. The pivot elements α_j are computed for each remaining spike and the largest, α_{max}, is determined. Let column s be the first spike such that α_s is larger than $T_c*\alpha_{max}$. Then columns k and s are interchanged.†

The above process amounts to Gaussian elimination with column interchanges.

Increasing T_r towards 1.0 will tend to

(a) increase the amount of column interchanging (spike-swapping);

(b) increase the total no. of spikes (if column k above is a triangle column the spike count will increase by 1);

(c) increase the density of the LU factorization (fill-in occurs only in spike columns);

(c) improve numerical stability.

Increasing T_c towards 1.0 will tend to

(a) increase (slightly) the density of the LU factors (early spikes are generally shorter and will have less fill-in than later spikes),

(b) improve numerical stability.

† If there is no suitable substitute, column k will be replaced by an appropriate slack vector.

For large-scale problems the total number of spikes is critical and a compromise must be reached between sparsity and stability. The default values for T_r and T_c are safe for most problems. If numerical difficulties arise, T_r can be increased to 0.01 and T_c to 0.9 or 0.99 without serious loss of efficiency. Larger values for T_r may degrade performance significantly.

For small dense problems (i.e., "non-LP" problems such as those arising from data-fitting), maximum stability is readily obtained by setting both T_r and T_c to 0.99 (say). The loss of efficiency will be negligible.

Stability during <u>updating</u> of the LU factorization is controlled by the tolerance T_m. In this implementation of the method of Bartels and Golub, maximum stability is obtained via the default value $T_m = 0.99$ with essentially <u>no loss of efficiency</u>. There is normally no reason to use a smaller tolerance. However, one can simulate the numerical properties of the method of Forrest and Tomlin (1972) by setting $T_m = 0.0$.

IX. System Information -- Machine-Dependent Matters

The source code for MINOS is approximately 8300 cards of standard Fortran IV. One of the design goals has been to maintain compatibility with the Fortran compilers on the principal large scientific machines (e.g., IBM, CDC, Univac, Burroughs). At the same time it has been possible to maintain a reasonably high level of efficiency and flexibility. For example, problems of arbitrary size may be solved without recompilation of the program, and the method for allocating storage can take maximum advantage of the available word sizes on a particular machine.

The main items involved in installing the system on a machine other than the IBM Systems 360 and 370 are:

--The setting of 4 variables in subroutine INITLZ.

--The hash function used in subroutine HASH.

--The file numbers used for the reader, printer and scratch files.

These topics are documented in the following sections.

IX.1. Precision, Word Sizes

Four scalar quantities must be initialized in subroutine INITLZ to match any particular machine. Their values depend on whether single or double precision floating-point arithmetic is required throughout, and their meaning is best understood in terms of the following values which are appropriate for the IBM System/370:

```
EPS    = 16.0**(-13)
NWORDR = 2
NWORDI = 2
NWORDH = 4
```

The variables in all relevant subroutines are implicitly typed as follows:

```
IMPLICIT    REAL(A-B), REAL*8(C-G, O-Z), INTEGER(I-N), INTEGER*2(H)
```

Furthermore, all array storage is provided by segmenting a single array of the form

```
REAL*8  Z(NWCORE)
```

For the IBM 370, the four critical values above are therefore interpreted as follows:

1. EPS contains the smallest positive number such that

   W = 1.0 + EPS

   gives W a value greater than 1.0. (We assume that the Fortran compiler will convert 1.0 to double precision and perform the addition in double precision.)

INSERT or LOAD file.

75

2. NWORDR = 2 means that if some array is of type REAL then 2 elements of that array occupy the same storage as one element of the REAL*8 array Z. Similarly, for arrays of type INTEGER and INTEGER*2, the respective numbers are 2 and 4 elements per word of Z.

For machines with suitable single precision and no half-word integers (e.g., CDC, Burroughs), REAL*8 and INTEGER*2 are replaced by REAL and INTEGER respectively, and the word-size indicators all take the value 1. The value of EPS should then correspond to *single precision.*

IX.2. Subroutine HASH

This routine is used for storing and searching a table of row names during input of the MPS file. Some machine-dependent "hash function" is required for converting any given 8-character name to a non-negative integer.

Specification

```
      SUBROUTINE HASH( LEN,NEN,NCOLL,
     1 KEY1,KEY2,MODE,KEYTAB,NAME1,NAME2,KA,FOUND )
      INTEGER    KEYTAB(LEN),NAME1(NEN),NAME2(NEN)
      LOGICAL    FOUND
```

Method

See Brent (1973).

Remarks

1. The first application of the required hash function is to the integer parameters KEY1 and KEY2. These variables contain the left and right halves of an 8-character row name, read under 2A4 format. Thus each integer contains 4 characters left-justified. If the resulting bit pattern for each is a valid integer (as is the case on the IBM 360 and 370) then the statement

```
KEY = IABS(IABS(KEY1) - IABS(KEY2))
```

defines a transformation which will produce a suitable non-negative integer KEY from the quantities KEY1 and KEY2. (Note that subtraction of non-negative integers cannot cause overflow.)

2. On some machines (e.g., the Burroughs B6700), reading the integer variable KEY1 (say) under A4 format will set bits which are reserved for the exponent in REAL variables. In such cases the word-length will be greater than 4 characters, and the simplest solution is to perform a right-shift of N bits, where N is chosen to clear the exponent field, e.g.,

```
K1  = IRSHFT( KEY1,N )
K2  = IRSHFT( KEY2,N )
KEY = IABS(K1-K2)
```

3. Note that KEY1 and KEY2 are later stored <u>without</u> transformation in the integer arrays NAME1 and NAME2. It must be possible to perform a test for equality of the form

```
IF (KEY1.EQ.NAME1(KT) .AND. KEY2.EQ.NAME2(KT)) GO TO 60
```

even if the bit pattern of any of the variables involved is not strictly that of an integer. (On the Burroughs B6700 it is safer to replace .EQ. by .IS.)

4. The same hash function must be used later within subroutine HASH to produce a non-negative integer KEY from the integer variables NAME1(KT) and NAME2(KT).

IX.3. Files 5, 6 and 8

The following file numbers are to some extent built in to the system:

1. File 5 -- the card reader.
   (a) The SPECS file is normally assumed to be file 5. It may be altered by changing the variable ISPECS in subroutine GO.
   (b) The MPS file default value is 5. It may be altered at run time by a SPECS card of the form

```
INPUT FILE      k
```

2. File 6 -- the printer.

All printed output is written on file 6. This includes listings of the SPECS and MPS data, parameters, iteration log, solution, error messages, etc. If necessary, the characters "WRITE(6" may be altered in all relevant subroutines.

3. File 8 -- the scratch file.

   (a) This file is used during input of the SPECS file. It facilitates interpreting of free-format data in a way that is machine-independent.

   (b) The scratch file is later used to store ROW and COLUMN names from the MPS data.

   (c) For both purposes the recording format should be fixed and blocked, with a logical record length of 8 characters (n.b. not 80). The block-size should be at least 800 or 1600 characters for efficiency.

   (d) The file number may be altered by changing the variable ISCRCH in subroutine GO. It must be different from the value in variable ISPECS.

Note that certain files are subject to a Fortran REWIND, except if their file numbers are 5 or 6 as the case may be.

IX.4. Using MINOS as a subroutine

As currently implemented, MINOS is intended for use primarily as a "stand-alone" system, which simply solves a sequence of problems and then terminates. Figure 5 illustrates the subroutine hierarchy.

AD-A044 906 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 9/2
MINOS. A LARGE-SCALE NONLINEAR PROGRAMMING SYSTEM (FOR PROBLEMS--ETC(U)
FEB 77 B A MURTAGH, M A SAUNDERS N00014-75-C-0865
UNCLASSIFIED SOL-77-9 NL

2 OF 2
AD
A044906

END
DATE
FILMED
11-77
DDC

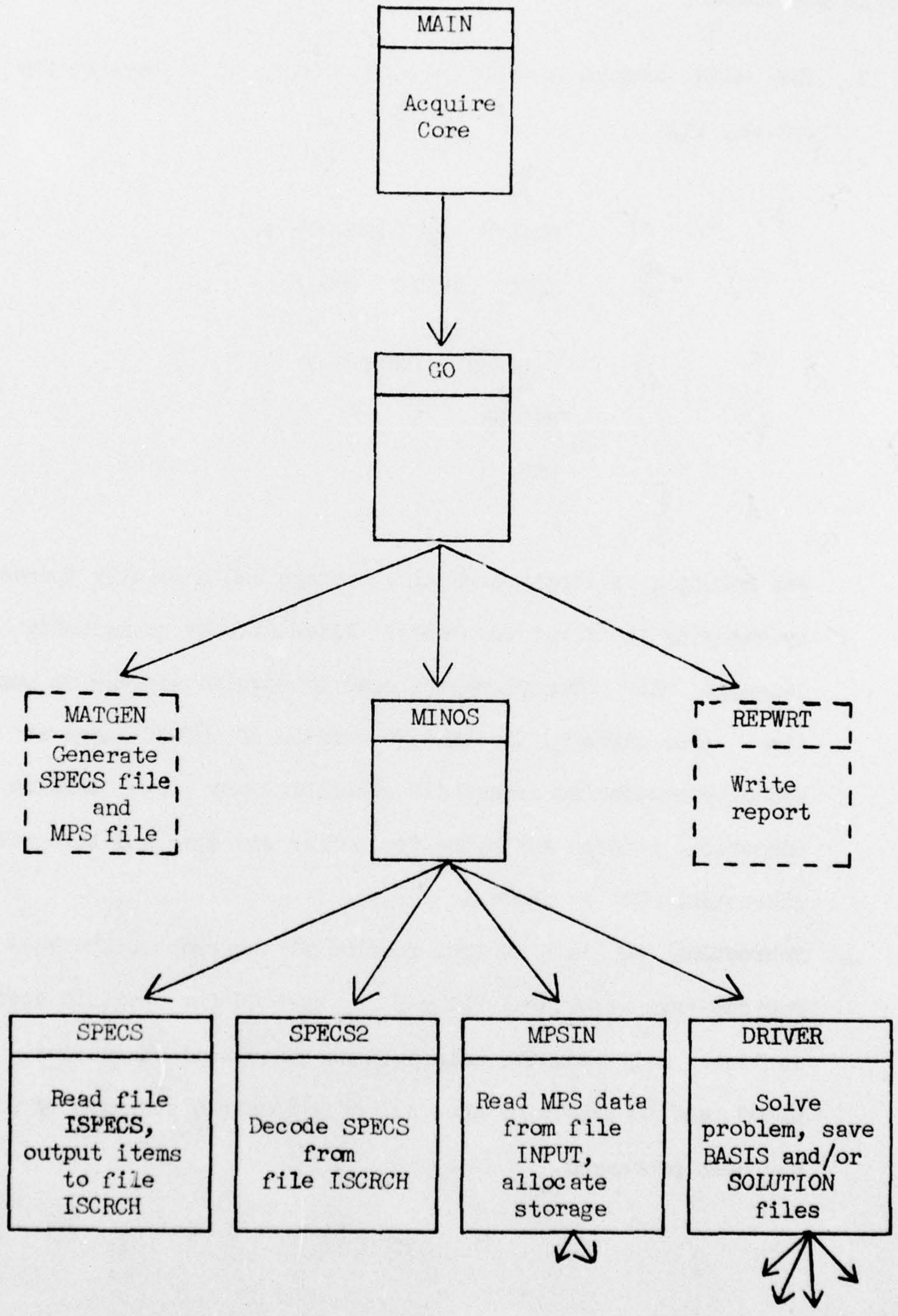
MAIN
Acquire Core
GO
MATGEN
Generate SPECS file and MPS file
MINOS
REPWRT
Write report
SPECS
Read file ISPECS, output items to file ISCRCH
SPECS2
Decode SPECS from file ISCRCH
MPSIN
Read MPS data from file INPUT, allocate storage
DRIVER
Solve problem, save BASIS and/or SOLUTION files

Figure 5. Subroutine hierarchy

In particular,

1. The MAIN program provides a single array of storage to the system, thus:

```
REAL*8      Z(20000)
DATA    NWCORE/20000/

CALL GO( Z,NWCORE )
RETURN
END
```

   For solution of larger problems, storage can be easily increased by changing the first two cards. Alternatively an assembly language MAIN program may be used to acquire storage at run-time. (For example, the IBM 370 version of MINOS uses an assembly routine to access all available core via a GETMAIN operation, returns 40K bytes for buffer storage, and then calls subroutine GO as above.)

2. **Subroutine** GO is a control routine at the root of the subroutine tree structure. It may be expanded for specific applications. As given, its main purpose is to call subroutine MINOS and to determine when all of a possible sequence of problems has been processed.

3. Subroutine MINOS and its sub-programs communicate with the outside world by means of certain files, including the SPECS, MPS, BASIS and SOLUTION files. Subroutine MINOS also has output parameters which define:

   (a) the stopping condition;

   (b) the dimensions of the problem that has just been processed;

   (c) the positions within array Z (above) where certain sub-arrays are stored. The latter contain the final values for the solution vector x, the dual variables π, and the state vector defining whether each variable x_j is basic, supe[illegible] or nonbasic at upper or lower bound.

The subroutine structure is sufficiently modular to allow use of MINOS in several alternative environments. The principal routine to be modified for a particular application is subroutine GO. It is envisaged that in all cases, run-time parameters and the constraint information will be input via the SPECS and MPS files (rather than by construction in-core).

As an example, the dotted lines in Figure 5 indicate how Matrix Generator and Report Writer modules could be incorporated. The sequence of actions

(a) Generate SPECS file and MPS file

(b) Solve

(c) Write report

could be repeated 1 or more times.

Alternatively it may be convenient to run the Matrix Generator as a separate program. This could generate a series of problem specifications on one SPECS file (since the SPECS file is not rewound after processing of the END card). The corresponding MPS data could be generated as one MPS file (with multiple RHS, RANGE or BOUND sets), or as several separate files.

The Report Writer may access solutions either externally (through the SOLUTION file) or in-core (through the parameter list of subroutine MINOS as noted above). In some cases the SCRATCH file may be useful to provide the ROW and COLUMN names as a sequence of 8-byte records.

A slightly different situation arises if MINOS is to be used iteratively for solving a sequence of linearly constrained problems in which the solution to one problem defines the constraints for the next. This is a possible approach to solving problems with nonlinear constraints. In such cases it is likely that a large portion of the constraint matrix will in fact be constant. The Matrix Generator could set this up as a permanent file, while generating the MPS and SPECS files required for the first Solve step. After each Solve, the "Report Writer" could then

(a) Access the solution as described above, test for convergence and stop if necessary.

(b) Generate new matrix coefficients and merge with the permanent file to produce a new MPS file.

(c) Generate an appropriately modified *SPECS* file.

(d) Modify relevant parameters used in the objective function.

(e) Return to the Solve step.

Although purely in-core operation is generally preferable, most of the computation for large problems will occur in the Solve step. The interspersed file-handling should therefore incur relatively negligible cost.

X. System Information -- IBM Version

This section illustrates the Job Control required to compile, run and modify MINOS on an IBM System 360 or 370 operating under OS. Some of the parameters used may be peculiar to OS/VS2 and/or the particular installation at the Stanford Linear Accelerator Center. Various DSNAMES and VOLUME names may also need to be altered. However, the main requirements should be apparent.

X.1. To Compile MINOS and Save the Resulting Load Module

In this example an executable program called NLPPGM is created as a member of the Partitioned Data Set called WYL.W8.MXS.LOAD, which will reside on a 2314 disk unit called WYLOOA.

The compile time for 8000 Fortran source cards using the IBM Fortran (H Extended) compiler with full optimization is approximately 40 seconds on an IBM 370/168.

```
//   JOB   ,TIME=(1,00),REGION=256K
//   EXEC   FORTHCL,PARM.FORT='OPT=2,NOMAP,NOSOURCE',
//          FORTPRT=DUMMY,
//          FORTTIM='(1,00)',FORTVER=NEW
//FORT.SYSIN  DD  *

      Fortran source

/*
//LKED.SYSLMOD  DD  DSN=WYL.W8.MXS.LOAD,UNIT=DISK,
//          DISP=(NEW,CATLG),VOL=SER=WYLOOA,
//          DCB=(RECFM=U,LRECL=3520,BLKSIZE=3520),
//          SPACE=(TRK,(40,5,1),,USE)
//LKED.SYSIN  DD  *
 ENTRY   MAIN
 NAME    NLPPGM
/*
```

X.2. To Run MINOS

The following job is suitable for running the load module created by the preceding example, or by the example in Section X.4. below. Note that the DDNAMES of the form 'FTnnF001' do not have the usual 'GO.' attached, because the EXEC card specifies NLPPGM directly rather than through a Cataloged Procedure.

The 3rd and 4th characters in each 'FTnnF001' DDNAME refer to file numbers which are used in Fortran I/O statements such as READ(nn, f).

FT05F001	Refers to the card reader.
FT06F001	Refers to the printer.
FT08F001	Defines the scratch file. Note that the logical record length is LRECL=8, not LRECL=80.
FT09F001	etc. refer to various basis and data files as required for a particular run. Note that DISP=SHR implies that a file already exists (as does DISP=OLD). If necessary some dummy file should be set up before the run. Otherwise, DISP=SHR must be replaced by DISP=(NEW,KEEP) and appropriate DCB and SPACE parameters must be defined.

```
//  JOB  ,TIME=(0,15),REGION=320K,MSGLEVEL=(0,0)
//MINOS  EXEC  PGM=NLPPGM
//STEPLIB   DD  DSN=WYL.W8.MXS.LOAD,DISP=SHR
//FT06F001  DD  SYSOUT=A,
//          DCB=(RECFM=FBA,LRECL=133,BLKSIZE=3458,BUFNO=2)
//FT08F001  DD  UNIT=SYSDA,SPACE=(TRK,(4,4)),
//          DCB=(RECFM=FB,LRECL=8,BLKSIZE=1600)
//FT09F001  DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.BASIS1,
//          DISP=SHR
//FT10F001  DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.BASIS2,
//          DISP=SHR
//FT11F001  DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.MPSDATA
//          DISP=SHR
//FT05F001  DD  *

     SPECS

     MPS data, if INPUT FILE 11 is not used  (FT11F001  above)

     Any data required by subroutine CALCFG

/*
```

X.3. <u>To Compile CALCFG and Other Subroutines and Link to MINOS</u>

The following job shows how to link one or more new or altered subroutines to MINOS and run the resulting program. This mode of operation is normally used to include a user's own nonlinear objective subroutine (CALCFG) or to test special versions of some of the subroutines in MINOS.

In the Link Edit step, the two XLIB cards refer to an existing load module. The new load module is not saved.

```
//  JOB  ,TIME=(0,15),REGION=320K,MSGLEVEL=(0,0)
//CLG  EXEC  FORTHCLG,PARM.FORT='OPT=2,NOMAP,NOSOURCE',
//           FORTPRT=DUMMY,LKEDPRT=DUMMY,
//           FORTTIM='(0,40)',FORTVER=NEW
//FORT.SYSIN  DD  *


     Fortran source for new subroutines


/*
//LKED.XLIB    DD  DSN=WYL.W8.MXS.LOAD,DISP=SHR
//LKED.SYSIN   DD  *
  INCLUDE XLIB(NLPPGM)
  ENTRY    MAIN
/*
//GO.FT08F001   DD  UNIT=SYSDA,SPACE=(TRK,(4,4)),
//              DCB=(RECFM=FB,LRECL=8,BLKSIZE=1600)
//GO.FT09F001   DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.BASIS1,
//              DISP=SHR
//GO.FT10F001   DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.BASIS2,
//              DISP=SHR
//GO.FT11F001   DD  UNIT=DISK,VOL=SER=SCR001,DSN=WYL.W8.MXS.MPSDATA,
//              DISP=SHR
//GO.SYSIN  DD  *


     SPECS

     MPS data, if INPUT FILE 11 is not used  (GO.FT11F001  above)

     Any data required by subroutine CALCFG


/*
```

X.4. <u>To Create and Save a New Load Module</u>

In this example the LKED.SYSLMOD card specifies where the new version of MINOS is to be saved. About 40 tracks of 2314 disk are required, or about 20 tracks of 3330 disk.

If necessary, any object decks (e.g., from assembly language versions of certain subroutines) should be included immediately after the IKED.SYSIN card.

```
//  JOB   ,TIME=(0,15),REGION=256K,MSGLEVEL=(0,0)
//CL    EXEC  FORTHCL,PARM.FORT='OPT=2,NOMAP,NOSOURCE',
//              FORTPRT=DUMMY,
//              LKEDPRT=DUMMY,
//              FORTTIM='(0,40)',FORTVER=NEW
//FORT.SYSIN  DD  *

      Fortran source for new subroutines


/*
//LKED.SYSLMOD  DD  DSN=WYL.W8.MXS.LOAD,UNIT=DISK,
//          DISP=(NEW,KEEP),VOL=SER=WYL00B,
//          DCB=(RECFM=U,LRECL=3520,BLKSIZE=3520),
//          SPACE=(TRK,(40,5,1),RLSE)
//LKED.XLIB   DD  DSN=WYL.W8.MXS.LOAD,DISP=SHR
//LKED.SYSIN  DD  *
 INCLUDE XLIB(NLPPGM)
 ENTRY   MAIN
 NAME    NLPPGM
/*
```

Note that:

--the old load module library (WYL.W8.MXS.LOAD) is assumed to be CATALOGed on the appropriate volume (e.g., WYLOOA as in the first job);

--the new load module library has the same name and must therefore reside on a different volume, in this case WYLOOB;

--the name of the new executable program is also the same as before (NLPPGM).

The purpose here is to make creation of the new version of MINOS transparent to the previous two jobs, which use whatever version happens to be referred to by the system CATALOG. It should now be possible to type a command of the form

```
RECATALOG  LOAD  ON  WYLOOB
```

on the local terminal system, and the deed will be done.

If the required terminal system facilities are not available, the following job illustrates how the RECATALOG operation may be accomplished using JCL alone. The point here is that the new load module can be cataloged only if the old version is uncataloged in a previous job step.

```
//   JOB   ,TIME=(0,15),REGION=256K,MSGLEVEL=(0,0)
//CL    EXEC   FORTHCL,PARM.FORT='OPT=2,NOMAP,NOSOURCE',
//             FORTPRT=DUMMY,
//             LKEDPRT=DUMMY,
//             FORTTIM='(0,40)',FORTVER=NEW
//FORT.UNCAT  DD  DSN=WYL.W8.MXS.LOAD,DISP=(SHR,UNCATLG)
//FORT.SYSIN  DD  *
```

```
     Fortran source for new subroutines

/*
//LKED.SYSLMOD  DD  DSN=WYL.W8.MXS.LOAD,UNIT=DISK,
//          DISP=(NEW,CATLG),VOL=SER=WYL00B,
//          DCB=(RECFM=U,LRECL=3520,BLKSIZE=3520),
//          SPACE=(TRK,(40,5,1),RLSE)
//LKED.XLIB    DD  DSN=WYL.W8.MXS.LOAD,DISP=SHR,
//              UNIT=DISK,VOL=REF=*.CL.FORT.UNCAT
//LKED.SYSIN   DD  *
  INCLUDE XLIB(NLPPGM)
  ENTRY   MAIN
  NAME    NLPPGM
/*
```

Beware that the old load module will be uncataloged even if the compile or link edit steps fail.

XI. Further Examples

XI.1. Example 2 -- An Unconstrained Optimization Problem

The following data illustrates minimization of the well known function (Rosenbrock, 1960)

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

without any general linear constraints.

Notes

1. At least one "constraint" is required in all problems. In this case the simplest technique is tc include a free (non-binding) row, called DUMMYROW here. The basis matrix will remain $B = 1$ throughout, corresponding to the slack variable on the free row.
2. The COLUMNS section contains just a list of the variable names, and the RHS section may be null.
3. The bounds $-10 \leq x_j \leq 10$ are included via the bound set BNDX, mainly as a matter of good practice. (Their presence does not imply additional work.) A uniform set of this kind could also be requested by the SPECS cards

```
LOWER BOUND     -10.0
UPPER BOUND      10.0
```

FUNCTION AND GRADIENT CALCULATION (for example 2)

```
      SUBROUTINE CALCFG( MODE,N,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),F,G(N)
C
C     -----------------------------
C     ROSENBROCK'S BANANA FUNCTION
C     -----------------------------
      S = X(2) - X(1)**2
      T = 1.0  - X(1)
      F = 100.0*S**2 + T**2
      G(1) = -400.0*S*X(1) - 2.0*T
      G(2) =  200.0*S
      RETURN
C
C     END OF CALCFG
      END
```

SPECS

```
BEGIN ROSENBROCK

   ITERATIONS                    100

   NONLINEAR VARIABLES             2
   SUPERBASICS                     2
   HESSIAN DIMENSION               2

   LINESEARCH TOLERANCE          0.1
END
```

MPS DATA

```
NAME          ROSENBROCK
ROWS
 N  DUMMYROW
COLUMNS
    X1
    X2
RHS
BOUNDS
 LO BNDX      X1              -10.0
 UP BNDX      X1               10.0
 LO BNDX      X2              -10.0
 UP BNDX      X2               10.0
 FX INITIAL   X1               -1.2
 FX INITIAL   X2                1.0
ENDATA
```

4. The INITIAL bound set illustrates how the starting point $(x_1, x_2) = (-1.2, 1.0)$ may be specified. Since the limit on SUPERBASICS is sufficiently high (2 here) both x_1 and x_2 will initially be superbasic at the specified values.

5. If the INITIAL bound set were not used, x_1 and x_2 would initially be nonbasic at the smaller of their bounds (in absolute magnitude), in this case the upper bounds 10.0.

6. If the user wished the variables to be genuinely unconstrained, the following bounds would have to be used:

```
BOUNDS
 FR FREE     X1
 FR FREE     X2
 FX INITIAL  X1          -1.2
 FX INITIAL  X2           1.0
```

where the INITIAL bounds are again optional. If the INITIAL bounds were absent, x_1 and x_2 would both be regarded initially as nonbasic at value zero.

XI.2. Example 3 -- A Quadratic Program

The following data illustrates solution of the quadratic program

$$\text{minimize} \quad \frac{1}{2} x^T Q x + c^T x$$

$$\text{subject to} \quad Ax \leq b, \qquad x \geq 0$$

FUNCTION AND GRADIENT CALCULATION (for example 3)

```
      SUBROUTINE CALCFG( MODE,N,X,F,G,NSTATE,NPROB )
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     X(N),F,G(N)
      COMMON     Q(3,3)
C
C     COMPUTATION OF  F = 1/2 X'*Q*X,     G = Q*X     FOR GIVEN  Q.
C     THE  COMMON  STATEMENT AND  SUBROUTINE SETQ  ARE PROBLEM
C     DEPENDENT.
C
C
      IF (NSTATE.EQ.1) CALL SETQ( Q,N )
C
      F = 0.0
      DO 300 I = 1, N
         S = 0.0
         DO 200 J = 1, N
            S = Q(I,J)*X(J) + S
  200    CONTINUE
         F    = X(I)*S + F
         G(I) = S
  300 CONTINUE
C
      F = 0.5*F
      RETURN
      END
```

SPECS

```
BEGIN QP
   ITERATIONS              50
   NONLINEAR VARIABLES      3
END
```

MPS DATA

```
NAME          QP
ROWS
 N C
 L A
COLUMNS
    X1        C              -8.0         A                 1.0
    X2        C              -6.0         A                 1.0
    X3        C              -4.0         A                 2.0
RHS
    B         A               3.0
ENDATA
```

for the particular data

$$Q = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 4 & 0 \\ 2 & 0 & 2 \end{bmatrix} \qquad c = \begin{bmatrix} -8 \\ -6 \\ -4 \end{bmatrix}$$

$$A = [\,1 \quad 1 \quad 2\,] \qquad b = \quad 3$$

Notes

1. The vector c is included as a free row in the constraint matrix, as in standard LP.
2. The $\leq$ shown above may of course include all forms of inequality as desired.
3. Often Q will not involve all variables. In such cases it is preferable to order the variables so that the nonzero rows and columns of Q are grouped together thus:

$$Q = \begin{bmatrix} \bar{Q} & \\ & 0 \end{bmatrix}$$

The number of NONLINEAR VARIABLES and the parameter N of subroutine CALCFG will then be the dimension of $\bar{Q}$.

4. It is assumed that Q is initialized by some subroutine SETQ during the first entry to subroutine CALCFG. The COMMON statement is also problem-dependent (and is included to ensure that Q retains its values for later entries). Otherwise subroutine CALCFG is suitable for any quadratic function involving the first N components of x.
5. Beware -- if $c \neq 0$, the factor 1/2 makes a vital difference to the function being optimized.
6. The optimal solution is

$$x = (1.3333,\ 0.7777,\ 0.4444)$$

$$\frac{1}{2}x^TQx = 8.2222$$

$$c^Tx = -17.1111$$

$$\text{objective} = -8.8888$$

XI.3. Example 4 -- Linear Least Squares

Data-fitting can give rise to a constrained least squares problem of the form

$$\underset{x}{\text{minimize}} \quad \|Xx - y\|_2$$

$$\text{subject to} \quad Ax \geq b, \qquad \ell \leq x \leq u.$$

This may be solved with MINOS by expressing the problem as

$$\underset{r,x}{\text{minimize}} \quad \frac{1}{2} r^T r$$

$$\text{subject to} \quad \begin{bmatrix} & A \\ I & X \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} \begin{matrix} \geq \\ = \end{matrix} \begin{bmatrix} b \\ y \end{bmatrix}$$

$$r \text{ "free,"} \qquad \ell \leq x \leq u \ .$$

Notes

1. As usual, $Ax \geq b$ may include all types of inequality.
2. $r = y - Xx$ is the "residual vector," and $r^T r$ is the "sum of squares."
3. The objective function and gradient are easily programmed as $f(r) = 1/2\ r^T r,\ g(r) = r.$
4. Use of MINOS in this context is certainly not claimed to be the most stable or efficient method known. Stability depends on the condition of a certain submatrix extracted from

$$\begin{bmatrix} A \\ X \end{bmatrix} .$$

If it is known that most of the variables x will be away from their bounds, efficiency could be improved by using the option (say). This would allow up to 10 variables at a time to be added to the set currently being optimized, instead of the usual 1.

XI.4. Example 5 -- The Discrete ℓ_1 Problem

An apparently similar data-fitting problem is

$$\underset{x}{\text{minimize}} \quad \|Xx - y\|_1$$

$$\text{subject to} \quad Ax \geq b$$

where $\|r\|_1 \equiv \sum_i |r_i|$. However, this is best solved by means of the following purely linear program:

$$\underset{\lambda,\mu}{\text{maximize}} \quad [\, y^T \quad b^T \,] \begin{bmatrix} \lambda \\ \mu \end{bmatrix}$$

$$\text{subject to} \quad [\, X^T \quad A^T \,] \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = 0 \,,$$

$$-1 \leq \lambda \leq 1, \qquad \mu \geq 0 \,.$$

Notes

1. The solution x is recovered as the "simplex multipliers."
2. The optimal value of $\|Xx - y\|_1$ is the sum of the absolute values of the "reduced costs" associated with λ. (It is also the maximal value of $y^T\lambda + b^T\mu$.)
3. If a particular row in A, b is required to be an equality constraint, the corresponding component of μ should be a free variable.
4. It does not appear simple to include the bounds $\ell \leq x \leq u$ except as part of $Ax \geq b$. In such cases it may be best to solve the original problem directly as a linear program, thus:

$$\underset{r,s}{\text{minimize}} \quad [\, e^T \quad e^T \,] \begin{bmatrix} r \\ s \end{bmatrix}$$

$$\text{subject to} \quad \begin{bmatrix} & & A \\ & & \\ I & -I & X \end{bmatrix} \begin{bmatrix} r \\ s \\ x \end{bmatrix} \begin{matrix} \geq \\ \\ = \end{matrix} \begin{bmatrix} b \\ \\ y \end{bmatrix},$$

$$r \geq 0, \qquad s \geq 0, \qquad \ell \leq x \leq u,$$

where $e^T = [\, 1\ 1 \cdots 1 \,]$.

XII. Conclusion

XII.1. A Word to the Linear Programmer

Those familiar with commercial mathematical programming systems will have noticed an absence of the usual algorithmic options (dual simplex method, GUB, parametrics, etc.), not to mention the normal wealth of data revision and file handling capabilities. However, it should be clear that MINOS is intended for use as an extension of such systems, not as a substitute. Certainly the necessary constraint data can be prepared and revised using standard matrix generators and the above-mentioned facilities. In fact, we would recommend that any sizeable model be thoroughly "debugged" as a purely linear program, using either MINOS or (preferably) a commercial system. An appropriate set of bounds can be used to fix the nonlinear variables at plausible values during this debugging stage. The final optimal solution thus obtained will generally provide a good starting point when the nonlinear objective is introduced and the bounds on nonlinear variables are relaxed. (In this context it is essential to follow the suggestions of Section VI.)

Note that if the constraint data has already been debugged, the INITIAL bounds set of Section III.2 allows this fixing and subsequent freeing of nonlinear variables to be accomplished automatically within a single run.

XII.2. To the Nonlinear Programmer

Nonlinear programming might reasonably be expected to include nonlinear constraints--another vital subject which has received no mention here. In answer to this point we can only say that there is something fundamentally special about linear constraints, namely that it is possible to stay on them with relative ease. In implementing MINOS we have shown that a significant class of large-scale linearly constrained problems can be solved with only moderately more effort than would be required by a linear program of the same dimensions. This situation changes dramatically with the slightest mention of nonlinear constraints (unless perhaps they are included as penalty terms in the objective--a strategy which normally leads to ill-conditioned problems and/or crude approximations to the solution).

To illustrate, the Generalized Reduced-Gradient method of Abadie and Carpentier (1969) is directly analogous to the reduced-gradient approach used in MINOS, but an order of magnitude more work is involved (for the more general problem) in maintaining feasibility with respect to the nonlinear constraints. Recently, a derivative of MINOS has been developed (MINOS/GRG, Jain, Lasdon and Saunders, 1976) with the intention of handling about 100 nonlinear constraints and several hundred sparse linear constraints. As might be expected, run times on significantly nonlinear problems have proved to be 5 or 10 times what would be typical if all constraints were linear. This is not to discourage use of the system on real problems, but it should serve to underscore the quantum difference between the two types of nonlinear program. Alternative implementations must be pursued to help narrow the gap.

In the meantime, we hope that for many users, the facilities now available in MINOS will provide a useful first step into the nonlinear world.

Acknowledgements

The authors are indebted to the following persons for their contributions (large or small, knowingly or otherwise) to the coding of MINOS: R.L. Asmuth, H.M. Barber, R.P. Brent, I.S. Duff, P.E. Gill, K.J. Morris, W. Murray, S.M. Picken, J.K. Reid, J.A. Tomlin, and M.H. Wright. The ideas of an INITIAL bounds set evolved from discussions with Arvind Jain and Leon Lasdon. Certain other facilities were influenced by the requirements and suggestions of Alan Manne and Richard Richels during production runs on the ETA Energy Model (Manne, 1976). Our thanks go also to John Tomlin and Margaret Wright for numerous discussions of implementation details, and again to Margaret Wright for many helpful comments on the manuscript and for preparing the very comprehensive index.

The groundwork for MINOS was laid while the authors were respectively at the Department of Information Science, Victoria University of Wellington, and the Applied Mathematics Division, D.S.I.R., Wellington. Subsequent development of the system was aided immeasurably by the computing facilities at the Stanford Linear Accelerator Center, particularly by the Wylbur/370 timesharing system and the Watfiv compiler.

References

J. Abadie and J. Carpentier (1969), "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," in Optimization, R. Fletcher, Ed., Academic Press, 37-47.

R.H. Bartels (1971), "A Stabilization of the Simplex Method," Num. Math. 16, 414-434.

R.H. Bartels and G.H. Golub (1969), "The Simplex Method of Linear Programming Using LU Decomposition," Comm. ACM 12, 266-268.

R.P. Brent (1973), "Reducing the Retrieval Time of Scatter Storage Techniques," Comm. ACM 16, 105-109.

C.G. Broyden (1972), "Quasi-Newton Methods," in: W. Murray, ed., Numerical Methods for Unconstrained Optimization (Academic Press, London and New York), 87-106.

G.B. Dantzig (1963), Linear Programming and Extensions (Princeton University Press, New Jersey).

W.C. Davidon (1959), "Variable Metric Method for Minimization," AEC Research and Development Report ANL-5990.

R. Fletcher and M.J.D. Powell (1963), "A Rapidly Convergent Descent Method for Minimization," Computer J. 6, 163-168.

R. Fletcher and C.M. Reeves (1964), "Function Minimization by Conjugate Gradients," Computer J. 7, 149-154.

J.J.H. Forrest and J.A. Tomlin (1972), "Updating Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method," Mathematical Programming 2, 263-278.

P.E. Gill, W. Murray, S.M. Picken, H.M. Barber and M.H. Wright (1976), Subroutines INSRCH and NEWPTC, Ref. No. E4/16/0/Fortran/02/76, NPL Algorithms Library, DNAC, National Physical Laboratory, Teddington. (Crown Copyright Reserved.)

E. Hellerman and D.C. Rarick (1971), "Reinversion with the Preassigned Pivot Procedure," Math. Prog. 1, 195-216.

E. Hellerman and D.C. Rarick (1972), "The Partitioned Preassigned Pivot Procedure," in: D.J. Rose and R.A. Willoughby, eds., Sparse Matrices and their Applications (Plenum Press, New York) 67-76.

D.M. Himmelblau (1972), Applied Nonlinear Programming (McGraw-Hill).

IBM Document No. H20-0476-2, "Mathematical Programming System/360 Version 2, Linear and Separable Programming-User's Manual," (IBM Corporation, New York).

IBM Document No. SH20-0968-1, "Mathematical Programming System-Extended (MPSX), and Generalized Upper Bounding (GUB)" (IBM Corporation, New York).

A. Jain, L.S. Lasdon and M.A. Saunders (1976), "An In-core Nonlinear Mathematical Programming System for Large Sparse Nonlinear Programs," presented at ORSA/TIMS joint national meeting, Miami, Florida.

A.S. Manne (1976), "ETA: A Model for Energy Technology Assessment," Bell Journal of Economics, Autumn 1976, 381-406.

B.A. Murtagh and M.A. Saunders (1976), "Nonlinear Programming for Large, Sparse Systems," Report SOL 76-15, Department of Operations Research, Stanford University, Stanford, California.

E. Polak and G. Ribiere (1969), "Note sur la Convergence de Methodes de Directions Conjugees," Revue Francaise Inf. Rech. Oper., 16RI, 35-43.

H.H. Rosenbrock (1960), "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer J. 3, 175-184.

A. Perry (1976), "An Improved Conjugate Gradient Algorithm," Technical note, Department of Decision Sciences, Graduate School of Management, Northwestern University, Evanston, Ill.

D.C. Rarick (1975), An improved pivot row selection procedure, implemented in the mathematical programming system MPS III, Management Science Systems, Rockville, Maryland.

M.A. Saunders (1976), "A Fast, Stable Implementation of the Simplex Method Using Bartels-Golub Updating," in: J.R. Bunch and D.J. Rose, eds., Sparse Matrix Computations (Academic Press, New York and London), 213-226.

P. Wolfe (1962), "The Reduced Gradient Method," unpublished manuscript, the RAND Corporation.

P. Wolfe (1967), "Methods of Nonlinear Programming," in: J. Abadie, ed., Nonlinear Programming (North-Holland, Amsterdam), 97-131.

Index

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 77-9 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MINOS: A Large-Scale Nonlinear Programming System (For Problems with Linear Constraints) User's Guide		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Bruce A. Murtagh and Michael A. Saunders		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0865 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL ✓ Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program, Code 434 Office of Naval Research Arlington, VA 22217		12. REPORT DATE February
		13. NUMBER OF PAGES 127
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Linearly constrained optimization; optimization; linear programming; nonlinear programming; sparse matrices; MINOS; mathematical software; large-scale optimization.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

MINOS is a Fortran program designed to minimize a linear or nonlinear function subject to linear constraints, where the constraint matrix is in general assumed to be large and sparse. The User's Guide contains an overview of the MINOS System, including descriptions of the theoretical algorithms as well as the details of implementation. The Guide also provides complete instructions for the use of MINOS, and illustrates the diversity of application by several examples.